AFRL-PR-WP-TR-2006-2198

IMPROVED MODELING TOOLS FOR HIGH SPEED REACTING FLOWS

R. Kielb
J. White
P. Eiseman

Pyrodyne, Inc.
11520 Rolling Hills Drive
Glenwood, MD 21738

SEPTEMBER 2006

Final Report for 29 March 2005 – 29 December 2005

THIS IS A SMALL BUSINESS INNOVATION RESEARCH (SBIR) PHASE I REPORT.

PROPULSION DIRECTORATE
AIR FORCE MATERIEL COMMAND
AIR FORCE RESEARCH LABORATORY
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7251

NOTICE AND SIGNATURE PAGE

//Signature//
DEAN R. EKLUND
Program Manager

//Signature//
PATRICIA D. PEARCE
Chief, Propulsion Technology Branch

//Signature//
JENNIFER M. HARALSON, LtCol, USAF
Deputy Chief, Aerospace Propulsion Division
Propulsion Directorate

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. REPORT DATE ***(DD-MM-YY)*** September 2006	**2. REPORT TYPE** Final	**3. DATES COVERED *(From - To)*** 03/29/2005 – 12/29/2005

4. TITLE AND SUBTITLE IMPROVED MODELING TOOLS FOR HIGH SPEED REACTING FLOWS	**5a. CONTRACT NUMBER** FA8650-05-M-2594
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 65502F
6. AUTHOR(S) R. Kielb J. White P. Eiseman	**5d. PROJECT NUMBER** 3005
	5e. TASK NUMBER PA
	5f. WORK UNIT NUMBER RN
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Pyrodyne, Inc. 11520 Rolling Hills Drive Glenwood, MD 21738	**8. PERFORMING ORGANIZATION REPORT NUMBER** PD-TR-001-2006
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Propulsion Directorate Air Force Research Laboratory Air Force Materiel Command Wright-Patterson AFB, OH 45433-7251	**10. SPONSORING/MONITORING AGENCY ACRONYM(S)** AFRL-PR-WP
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) AFRL-PR-WP-TR-2006-2198

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
This is a Small Business Innovation Research (SBIR) Phase I Report. Report contains color.
PAO case number: AFRL/WS 06-1815; Date cleared: 26 Jul 2006.

14. ABSTRACT
This report was developed under SBIR contract for Topic AF05-194.

The objective of this work was to develop and demonstrate a means for the efficient integration of detailed numerical analysis into the design-optimization process. An integrated design optimization and engineering analysis tool has been demonstrated. This tool leverages the significant work required to set-up a detailed numerical analysis. The additional work required to execute the optimization includes formally defining the figure of merits, setting up the DAKOTA input file and possibly re-defining the geometry and topology for parametric grid generation. Setting up the DAKOTA input file is a straight forward task. Since the process requires a formal definition of the figure of merits, the system documents itself and is very repeatable. Including grid generation in the loop requires some additional effort, however, the potential gains in component performance and system operability are substantial.

15. SUBJECT TERMS
SBIR Report, design, optimization, CFD, design tool, grid generation

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT:**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** (Monitor)
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	44	Dean R. Eklund **19b. TELEPHONE NUMBER** *(Include Area Code)* N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

PYRODYNE, INC.
12310 Hungerford Manor Ct.
Monrovia, MD 21770
301-865-4211

June 30, 2006

Dr. Dean R. Eklund
AFRL/PRAT
Aerospace Propulsion Division
1950 Fifth St.
WPAFB, OH 45433-7251

Ref: Phase I SBIR Contract # FA8650-05-2594

Dear Dr. Eklund:

Pyrodyne hereby releases all data rights for the referenced contract and authorizes public release of our reports.

Sincerely,

Linda a Baumler

Linda A. Baumler
Treasurer/Business Manager

INTRODUCTION / OBJECTIVE

The objective of this work was to develop and demonstrate a means for the efficient integration of detailed numerical analysis into the design-optimization process. In this initial phase of the effort, the goal was to demonstrate the tools and techniques that allow for surface definition and grid generation to be manipulated inside an optimization loop without user intervention.

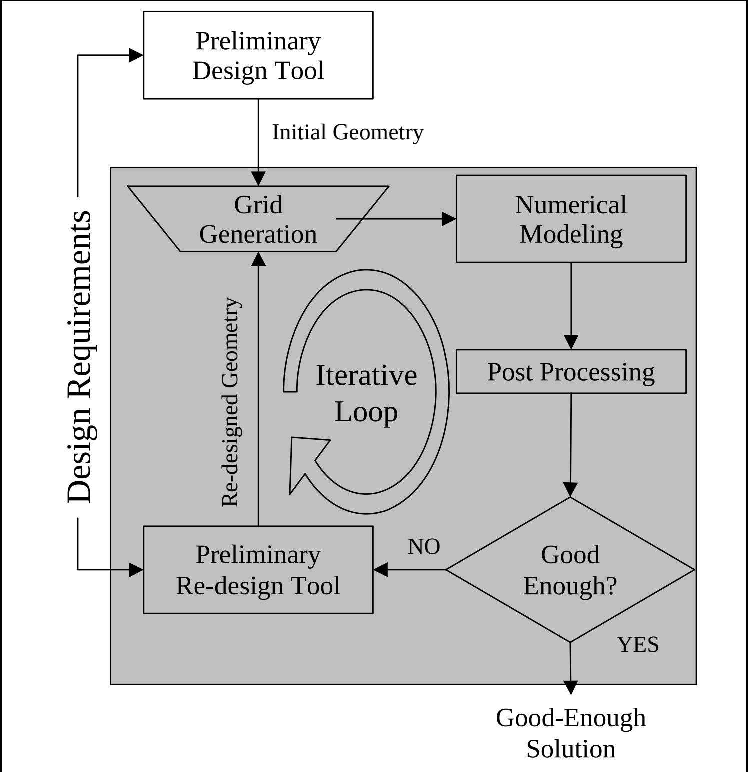


Figure 1, The Current Design Process

Often, substantial effort is required to set up a detailed numerical analysis. The overall goal of this work was to allow a user to leverage this effort by providing a generic interface to advanced tools, such as optimization, uncertainty analysis, trade studies, parametric modeling and the like.

The following example illustrates how this approach impacts the design process. A flow chart of the typical integration of numerical analysis into the design process is shown in Figure 1, the current design process. In this scenario, design requirements are passed to an engineer/engineering company whose job it is to design a part. The engineer generates an initial concept based on experience and available first-order tools. The resulting design is then numerically modeled and tested against the design requirements. If the design meets requirements, the job is done. If not, or if the design is for a competition, the engineer starts making trades to the original concept in an attempt to generate a design that meets requirements. In general, two passes through the detailed analysis loop are all that time and budget constraints will allow.

Figure 2 shows the optimized-design process. In this scenario, the engineer is given and/or generates a set of design constraints. This information is used to determine the design space. The engineer must also generate a set of Figures of Merit (FOMs). The objective function to be optimized is a combination of the FOMs. The relative weight of each FOM can, for example, be determined by system-level sensitivity studies. The result of this process is an optimized design with potentially several hundred passes

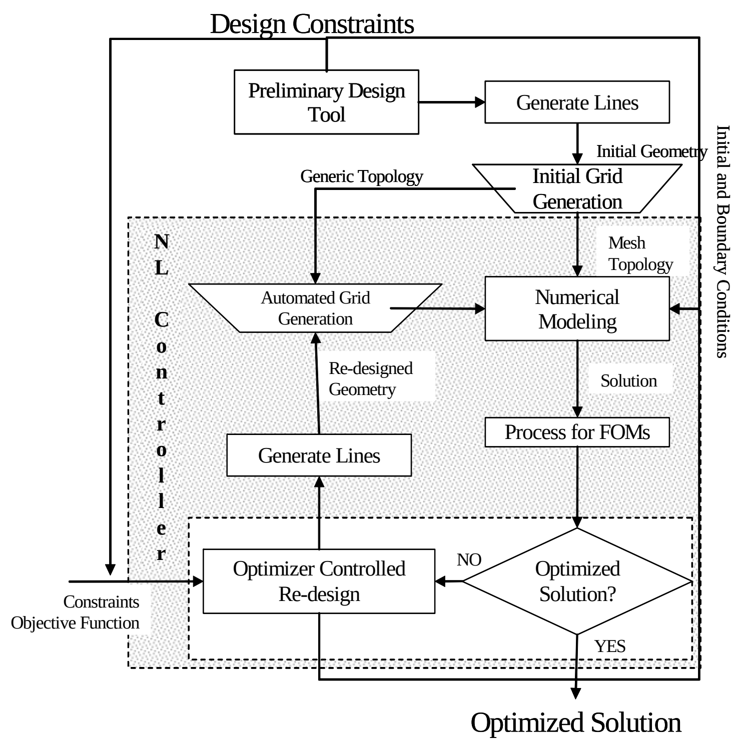


Figure 2, The Desired Design Process

through the detailed analysis loop. Additionally, a byproduct of the required set-up and mathematical rigor of the optimization procedure is that the process has documented the resulting solution.

The work performed under this effort can be broken up into three main parts; initial set-up, code integration and system demonstration. The initial set-up includes hardware selection and installation of off-the-shelf software. The code integration effort was primarily software development; scripts were written to link components of the analysis together and code was developed to generate the surfaces for the analysis. The software is generally in the form of stand-alone utilities but did, in certain cases, work its way into the main program architecture as an option. Finally, the system was tested using two sample cases, a jet in cross flow and a shape transition, both of which are described in detail later in the text.

INITIAL SET-UP

This work centered on putting the tools in place and operating them as a single system on the Beowulf cluster which was purposely built by Blue Blanket LLC (BBLLC) for this task. The tools used are VULCAN-CFD[1], DAKOTA[2] and GridPro[4]. VULCAN-CFD and DAKOTA are tools developed under government funding. DAKOTA is freely available, developed under the GNU public license. VULCAN-CFD is available to US citizens through NASA Langley Research Center (LaRC). GridPro is a commercial tool, available from the Program Development Company (PDC).

Computational Cluster

An eight processor cluster was leased from BBLLC with an initial service date of May 15. Although the hardware for the compute nodes specified (AMD 2800+ Barton core, 1GB RAM, gigabit ethernet network connection) was available within the initial time frame, the high capacity (one tera-byte) hot-swappable RAID 5 required on the front-end was not. This delayed initial service date by two weeks, to May 30. Once the system came on-line, it performed well throughout the seven month effort.

Software Installation

Once this cluster was in place, the off-the-shelf software was installed and tested. GridPro and DAKOTA were installed from binary distributions and the install went very smoothly. Since GridPro is a commercial product, it required the installation of a license manager and a license key. PDC provided no-cost copy of GridPro for this development work.

Since the VULCAN distribution is required to be compiled from source, the installation was a little more involved and requires a FORTRAN 90 compiler. The Portland Group (PGI) compiler, along with its license manager and key, was installed on the cluster and used for this purpose. To compile VULCAN, the MPI libraries had to be re-built, using the PGI compiler. These libraries could then be linked into the VULCAN executable. This executable was turned into an RPM and distributed to the compute nodes, along with a required PGI library. Installing VULCAN on the compute nodes allows for faster job start-up and less start-up network traffic relative to launching over a mounted network drive. The fact that this work was completed in a timely fashion is due in large part to the help of Dr. Robert Baurle at NASA LaRC and Mr. Kevin Fraze at BBLLC. Their work and guidance in supporting this effort is greatly appreciated.

CODE INTEGRATION

Once the off-the-shelf codes were installed and tested, the process of code integration could begin. The first step was to develop a strategy of how the optimization process will be executed on the cluster. It was a given that the VULCAN-CFD code will be run in parallel via jobs being submitted to the queuing system. DAKOTA can be run in either serial or parallel mode. For the purposes of this work, it was decided to run DAKOTA in serial mode on the front-end (which is not used for parallel computing). If DAKOTA were dealing with data points in the tens of thousands, it would be prudent to reconsider this choice. For the planned cases, where the number of function evaluations will be less than 100, the processing time required by DAKOTA is minimal, so running on the front-end seemed to be the appropriate choice.

GridPro can only be run serially, but it can be run either on a compute node or on the front-end. It was decided to node lock a license to the front-end, mainly because, from a cost perspective, this is the way many users would set up the system. Running GridPro on the compute nodes requires a server license and limits number of concurrent grid generation jobs to the number of licenses. Running GridPro node locked to a computer allows multiple grid generation jobs to be launched simultaneously, albeit slowing the solution. Additionally, this choice reduced the over head in writing the scripts that run the simulation.

Cluster Architecture

The cluster uses a Red-Hat based distribution called ROCKS[3], developed at the University of California San Diego (UCSD) under an NSF grant. The BBLLC cluster uses the Torque/MAUI combination to schedule jobs and manage resources and the

Ganglia cluster monitoring tool. This combination of tools worked very well, resulting in a cluster that was easy to use and monitor.

GridPro

Although there is a GUI available, GridPro was designed from the ground up to run from a command line interface, as well as through the GUI. This capability, along with its particular grid generation paradigm, allows it to be used in a “hands-off” manner via the Topology Input Language (TIL). Documentation for features can be found in the GP User’s Guide and Reference Manual[4].

PDC delivered two stand-alone utilities in support of the optimization cases, one for each case. The initial utility delivered creates a “cap” surface at the intersection of two arbitrary surfaces. The cap-surface is then used as a control surface during the grid generation process. The second utility delivered is a shape transition routine allowing for the generation of a transition section between a rectangular/oval/round duct to a rectangular/oval/round duct of different shape and/or cross-sectional area. The utilities are detailed in each of the sections describing the work where they were used.

VULCAN

A number of modifications were made to the VULCAN-CFD code during the Phase I effort. These modifications can be categorized as:

- Integration with GridPro.
- Integration with DAKOTA
- Improve load balancing
- Implementation of a new boundary condition

Integration with GridPro

The **GridPro** code produces C(0) multi-block structured grids. The design of GridPro is such that the grids it produces can consist of up to hundreds and occasionally even thousands of blocks. This plethora of blocks, which are useful in capturing geometric features and load balancing, makes manually specifying the boundary conditions and block-to-block connectivity of these grids utterly impractical. However, GridPro provides a method to graphically specify the boundary conditions by associating them with surfaces as well as automatically keeping track of and recording the block-to-block connectivity as the grid topology is generated. The grid connectivity and boundary conditions can then be exported into a formatted text file. The file, which is unique to GridPro, required that a translator was written that converts the information in the GridPro file into a form readable by the VULCAN-CFD code as well as the VULCAN-CFD code input file graphical user interface.

Improve load balancing

The large number of blocks produced by the GridPro code could also cause difficulties for the native VULCAN-CFD code load balancing algorithm that could result in poor performance on multi-processor platforms. These difficulties are due to the native algorithm not considering communication overhead when distributing the blocks among the available processors. This algorithmic deficiency could and has caused blocks to be distributed such that there would be much more inter-processor communication than need be. This could produce poor scaling and performance on parallel machines, particularly on machines with high inter-processor latency. To alleviate this deficiency a strategy in use by the unstructured grid community was adapted for use with structured multi-block grids. The graph partitioning code METIS developed at the University of Minnesota has been used to partition the multi-block grid in a manner similar to how unstructured grid are partitioned using METIS. The METIS code requires a graph of the computational grid to be partitioned where the graph is made up of vertices and edges. For unstructured grids there is a one-to-one correspondence between the graph vertices and the unstructured grid nodes and between the graph edges and the unstructured grid lines. However, the correspondence between the graph entities and multi-block structured grids is a little more abstract. Fortunately, it is possible to define a correspondence between the graph vertices and a structured grid block and between the graph edges and the structured grid block-to-block connectivity. This correspondence was exploited to develop a structured grid graphing subroutine that was implemented into the VULCAN-CFD code and the METIS code was integrated into the VULCAN-CFD code scripting to allow the code to be automatically run to produce graphs of multi-block structured grids, partition the graph, and use the resulting graph partition file to distribute the grids over the processors so as to simultaneously balance the computational load and minimize the communication overhead.

Integration with DAKOTA

In order to stream line the optimization analysis process, it was decided to modify the VULCAN-CFD to process the required Figure of Merit (FOM) data from the simulation output. These data were then read from the output files and passed back to DAKOTA which are then used to drive the optimization process. A number of FOMs already existed in the VULCAN-CFD code, most of which related to integrated forces and moments. The existing surface integration routines output data on an iteration-by-iteration basis. Several new parameters were added to the VULCAN-CFD code to support the fuel injector and the transition duct demonstration cases. In addition the FOM surface integration logic was extended from the original post-processing approach to include a real time approach. This was done to provide a real time output of the FOM to a separate file to simplify the process of interfacing VULCAN-CFD with IDEA-NL and to allow the IDEA-NL a way to monitor the CFD simulation in real time so that “converged” jobs could be detected and stopped and “divergent” jobs or jobs with poor FOM convergence could be detected and culled. The following FOMs are currently available in the VULCAN-CFD code (new additions in italics):

- Forces and moments
- Mass flow error

- Total heat flux
- *Fuel penetration height*
- *Fuel mixing efficiency*
- *Mass averaged total pressure loss*

Near real time plotting of the various output parameters was also added to the VULCAN-CFD GUI using Tecplot layout files and scripts. Figure 3 presents hydrogen (H_2) mass fraction contour for the sonic normal injection of hydrogen into a supersonic crossflow of air that was used to test the real time output of the parameters. Figure 4 presents plots of the convergence and figures of merit, including the fuel penetration height and mixing efficiency, added during the Phase I effort, for the aforementioned fuel injection demonstration case.

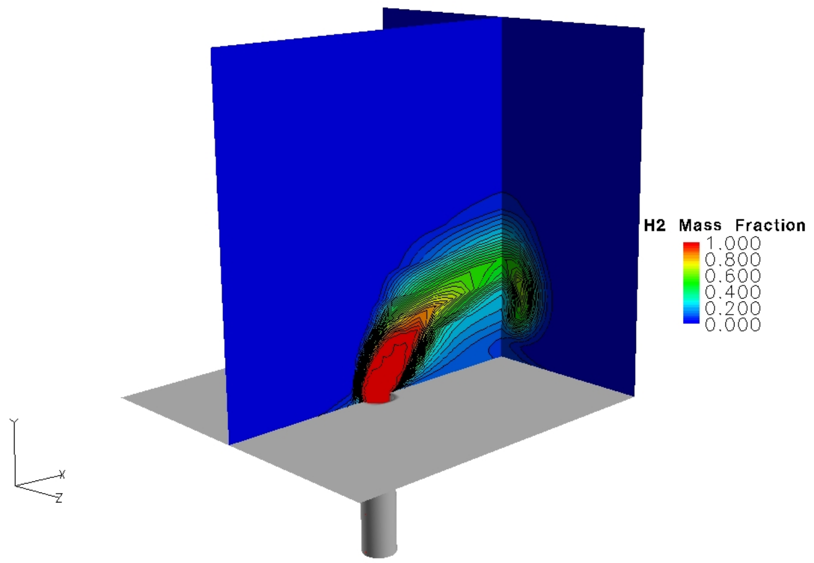


Figure 3, Contours of Hydrogen Mass Fraction on the Centerline and Outflow Planes

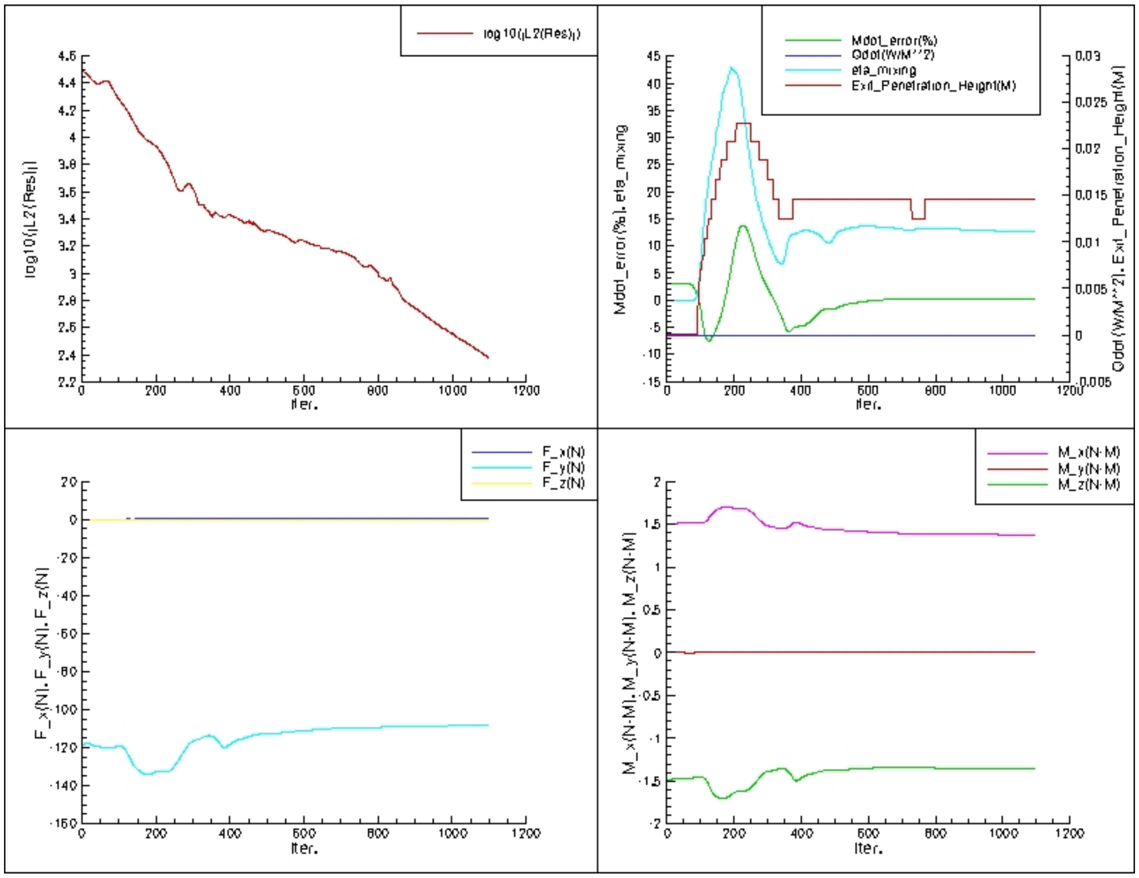


Figure 4, Real-Time Convergence and Various Parameters Including Fuel Mixing Efficiency and Penetration Height

Boundary Condition Modification

One final modification was made to the VULCAN-CFD code to streamline setting up the analysis of the fuel injector demonstration case. The existing specified mass flow/unit area boundary condition in VULCAN-CFD code was modified to allow specification of both the mass flow/unit area and the flow cross sectional area of the “baseline” grid. This was done to allow the total mass flow to be held constant as the cross-sectional area varied from geometry to geometry during the optimization process. This could have been done, with considerable effort, by adding a constraints, calculating cross sectional areas and modifying the VULCAN-CFD input file each system-level iteration. Adding the capability directly to VULCAN-CFD code was more efficient to implement, a benefit of having the source code available.

DAKOTA

As part of its distribution, DAKOTA provides a set of utilities to integrate various pieces of software via input/output files. This capability, largely provided by a suite of PERL scripts, was used during the phase I effort. The work-horse script is called *dprepro.prl*. It processes a file by searching for user-defined place holders and replacing them with user supplied data. The syntax is shown in Appendix 1.A, a listing of a driver script where the DAKOTA pre-processor utility was used.

TOOL DEMONSTRATION

FUEL INJECTOR CASE

Baseline

Starting an optimized analysis was very much similar to starting any other CFD analysis. A computational domain was defined and the appropriate type of analysis to be run identified. In this case, the computational domain was a one inch cube with a round tube 0.080 inches in diameter intersecting the cube at 90 degrees. This is shown Figure 5, Grid for Fuel Injector Study. The grid employs a wall spacing of 0.003 inches and a stretching factor of 1.1, so it is suitable for viscous calculations. This resulted in a total of 190,260 cells in the grid.

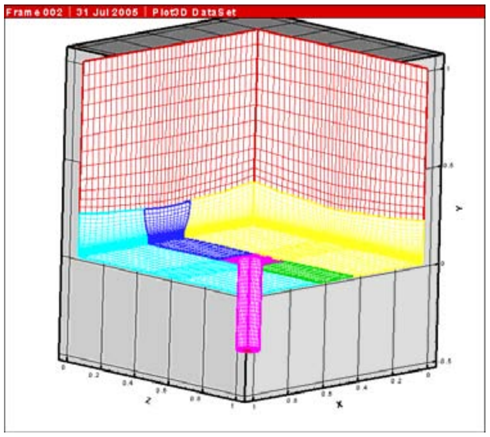


Figure 5, Grid for Fuel Injector Study

- 1 x 1 x 1 Cube
- 0.080 Dia hole
- Total Grid Points – 190,260 cells
- Wall spacing 0.003 spacing, stretch = 1.1

The purpose of this work is to demonstrate the tool and methodology, not to develop a fuel injector design. This fact, in combination with the limited computational resources, drove our choice of modeling detail. For the fuel injector case, jet penetration was chosen as the primary FOM. Other FOMs considered were not used for various reasons. For instance, a mixing efficiency would require a longer duct to see significant differences and combustion efficiency would add the requirement of modeling chemically reacting flow. Choosing penetration as our FOM simplified the required analysis, allowing the simulations to be run adiabatic with thermally perfect, frozen chemistry. The boundary condition types set on the simulation are shown in Figure 6.

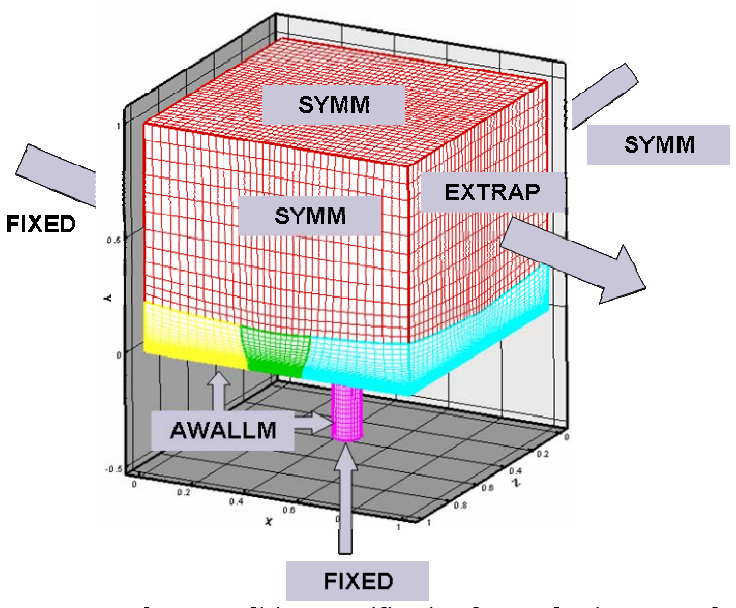


Figure 6, Boundary Condition Specification for Fuel Injector Study

The boundaries on three sides of the cube were set to symmetry; the outflow boundary is set to extrapolate; the wall boundaries set to adiabatic with wall functions; and, the inflow boundaries, both the main flow and fuel injector, were set to the fixed BC. Setting the fuel injector BC to be fixed instead of subsonic inflow was done to facilitate the initial optimization testing on a local workstation. This is further described in a section to follow. The fixed main inflow was set to be air incoming at Mach 2 flow. A VULCAN-CFD input file, which is located in the Appendix 1.B, was created for this case and the baseline fuel injector case was run.

Once the baseline simulation was complete, the optimization runs were started. The initial optimization runs were performed using only VULCAN-CFD and DAKOTA on a local workstation while the cluster was coming up. This was done to get some experience setting up a DAKOTA input file and develop a feel for how DAKOTA works. DAKOTA was given a fuel density range and set-up to maximize the fuel flow rate using the fuel density as the control variable. A DAKOTA driver script was written to modify a special VULCAN-CFD input template file where the tag $DENS$ was inserted in place of the values of the fuel density in the baseline input file. The script reads in the template input file, searches for the tag and replaces it with the value of density determined supplied to the script by DAKOTA. The script then parsed the VULCAN output file for the flow rate at the fuel injector inflow boundary which was then passed to DAKOTA as the value to optimize. This was the aforementioned motivation for running with the FIXED boundary on the fuel injector. Since the boundary condition under optimization is being set directly, the simulations only need to run a single iteration before they can be processed for the FOM. For subsequent runs where the figures of merit require a converged solution, the BC for the fuel injector is switched to SUBIN.

As expected, DAKOTA chose the maximum total density as the value of the independent variable that maximized the injector flow rate. No geometry changes within the optimization loop were attempted for this initial work.

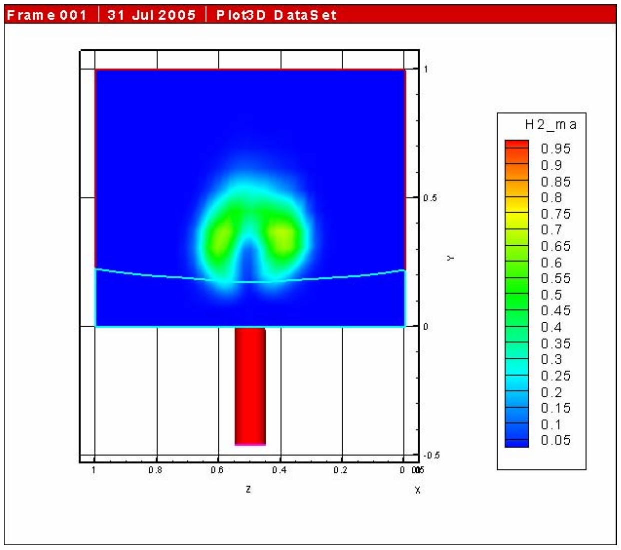


Figure 7, Hydrogen Plume at Exit Boundary at the Exit of the Computational Domain

The maximum density case was then run to convergence for use as the baseline case. This was an oversight as the equivalence ratio for this case was a little greater than five. This did lead to excellent penetration and plume growth, as can be seen in Figure 7. The plume is penetrated about half an inch and has spread to approximately 0.4 inches in diameter. Unfortunately, the high mass flow being injected caused the flow to shock down to subsonic in a large portion of the computational domain. This operation serves to highlight the fact that care must be given when setting up the domain to be optimized so that 100 of cpu-months are not spent to determine a solution which is not feasible for the operational system.

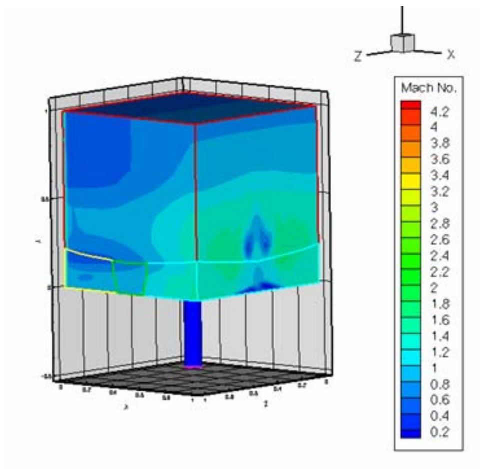


Figure 8, Mach Number at the Boundaries of the Computation Domain for High Phi Case

The flow rate was then dropped to correspond to a more reasonable phi of 0.5. This was used as the baseline case for the fuel injector optimization

GEOMETRY and GRID GENERATION

For the most part, the geometry required to generate the grid can be created internally in GridPro, making it convenient for the man-out-of-loop optimization. This is true of the cube (a set of 6 planes), the fuel injector tube (a super-ellipse) and the plane that defines the inflow boundary of the fuel injector tube. However, in order to generate an accurate grid, an internal "cap" surface must be created. This surface, which is created from the intersection of the fuel injector tube and plane, is shown in Figure 9. The purpose of the surface is to ensure the grid is held tightly (sharp corners) to the intersection of the fuel injector tube and the plane. This cap surface was only able to be generated through the GridPro GUI.

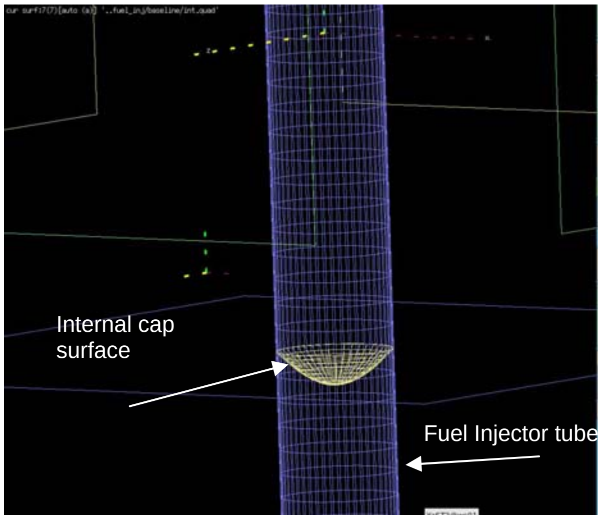


Figure 9, Fuel Injector "Cap" Surface

While the cluster was coming up, GridPro reworked the utility so that it could be called from the command line. The utility was completed and delivered during the initial two month period. The utility takes as input the coefficients and exponents of the super-ellipse for the fuel injector definition. The intersecting plane is defined by inputting a point on the plane and the normal to the plane. The final input to the utility is the name of the output file the surface is written to.

For reference, the equation of a super-ellipse is shown in Equation 1.

$$f(x,y,z) = \left| x/u \right|^{EXP} + \left| y/v \right|^{EXP} + \left| z/w \right|^{EXP} - 1 \qquad \textbf{Equation 1}$$

In order to run the optimization, *u*, *v* and EXP were chosen as the design variables. The intersecting plane was defined to be normal to the y-direction and pass through the origin. The y coefficient is set to be very large compared to the x and z coefficients, which yields the desired tube-like shape. Varying the x and z coefficients morphs the cross section of the fuel injector from a circle to an ellipse. Varying the exponent morphs the cross section of the fuel injector from a circle to a square. Other forms of the super-ellipse equation allow for a wider range of possible configurations. These cases would, however, require a more sophisticated topology.

Table 1, Values of the Design Variables Chosen for Optimization

	u	*w*	EXP
Nominal	20	20	2
Maximum	40	40	2
Minimum	10	10	15

The value ranges for the design variables are shown in Table 1. No effort was made to constrain the combinations of variables to conserve area.

RUN SCRIPT

Once the utility was complete, a run script was created that took input from DAKOTA, generated a cap surface and matching fuel injector tube, then a computational grid. The simulation is then run on the grid and the FOMs passed back to DAKOTA. The run script is discussed in detail in Appendix 1.C. The results of the analysis are discussed below.

RESULTS

As described above, this work allowed for the fuel injector cross section area to very from a circle through an oval to a square. The results of the analysis, after 58 iterations, are shown below in Figure 10.

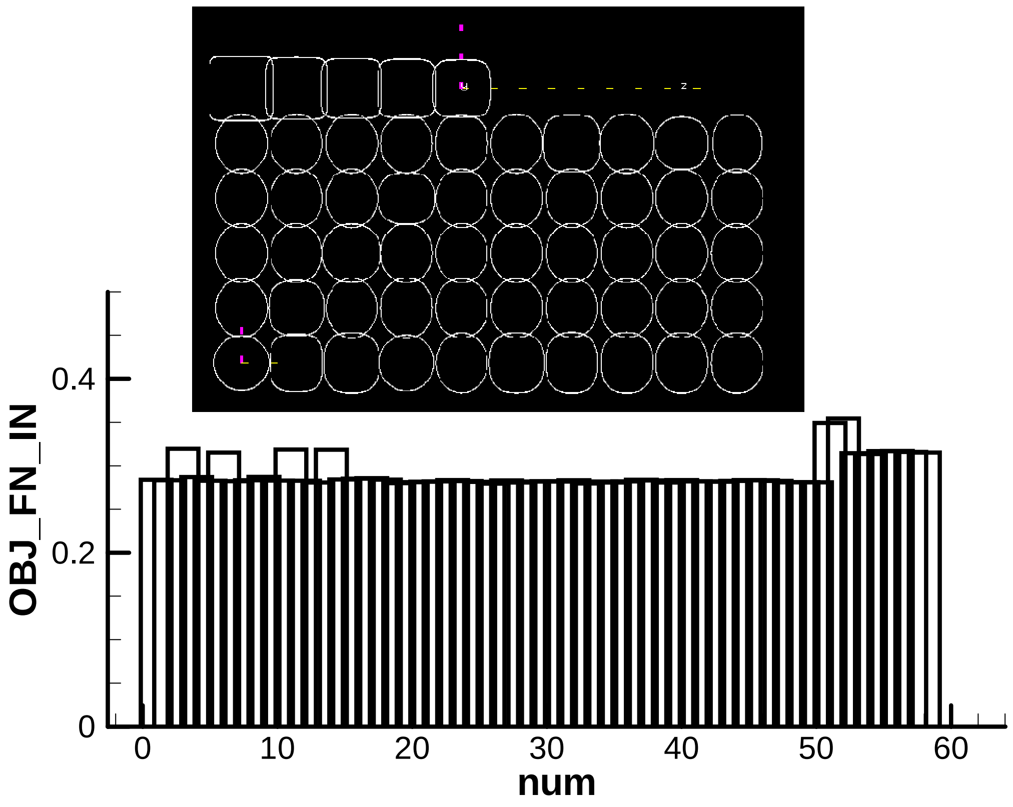


Figure 10, Results of the Fuel Injector Cross Section Optimization

Note that for approximately thirty iterations, the penetration height (the objective function) bounces around a minimum. Investigation revealed that the optimization was run to minimize the penetration height, as opposed to maximize it as was the stated goal. Further investigation showed the penetration height was a very strong function of mass flow and the solution bounced around the minimum area. The available VULCAN-CFD boundary conditions only allowed the mass flow per unit area to be set, not the absolute mass flow. While this could be overcome with significant effort using scripts to calculate areas and perform density ratios, the far easier solution was to add the capability into the VULCAN-CFD code. This was accomplished by modifying the SUBIN boundary condition as described in the Boundary Condition Modification section above.

The decision was made to begin the shape transition work while the new boundary condition was being implemented. Unfortunately, the remaining time was used working with the shape transition analysis, so fuel injector the analysis was not re-run with the constant mass flow boundary condition.

Shape Transition

In order to demonstrate the optimization of a shape transition, a generic transition, from a rectangle to a circle, was chosen. The rectangle, which is the inflow boundary, has an aspect ratio of four-to-one. The circle has an area equal to 95% of the area of the

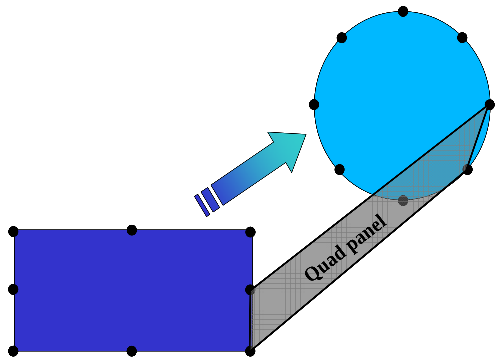


Figure 11, Schematic of Shape Transition with Control Fixed Points Shown

rectangle. For a rectangle with sides of 1 inch and 4 inches, the radius of the circle is 1.1 inches. The length of the transition section was set at one diameter, or 2.2 inches. As shown in Figure 11, eight control points were located around both the rectangular and circular cross sections. These points are frozen in the analysis, freezing the shape of the boundaries. These points, along with a ring of control points located mid-span, were used to control the geometry of the shape transition duct during the optimization.

Baseline Case

In order to generate a representative baseline, a surface was generated in a similar manner to what is currently done in industry. That is, to generate a lofted surface using a commercial CAD package. In this case, Solid Edge was used to generate the lofted transition duct. This duct, after being converted to the *STL* file format, is shown in Figure 12. Care must be taken when doing the file conversion as to capture the contour of the surface being manipulated.

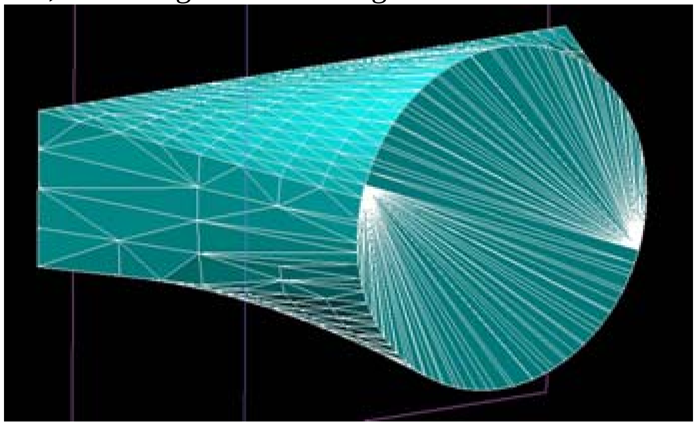
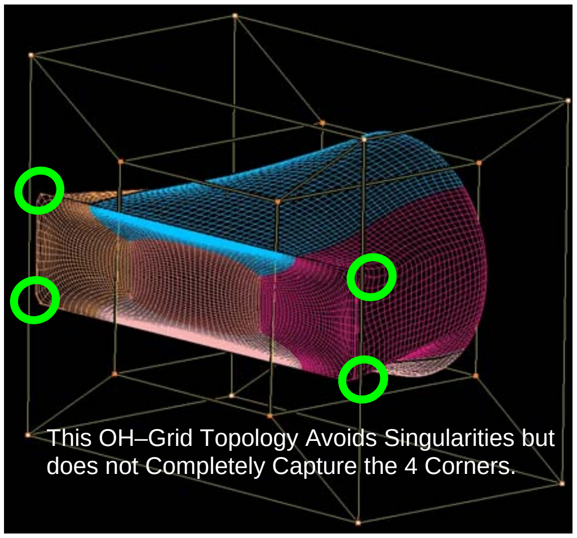


Figure 12. Baseline Lofted Surface Converted to an STL.

Once the surface is converted to an STL file format, it can be read into GridPro as a surface. Two additional grid control surfaces, planes to cap the inflow (rectangular) and outflow (circular) boundaries, are generated within GridPro.

Grid Generation

The topology laid out for the grid generation, along with the resulting grid, is shown in Figure 13. This topology generates an OH grid. While this grid works well for circular cross sections, additional steps must be taken to ensure the sharp corners in the rectangles are properly modeled. It was decided that, for the purpose of demonstrating the tool, the added complexity of capturing the sharp corners was not required. The affected areas are highlighted in the left hand picture of Figure 13.


This OH–Grid Topology Avoids Singularities but does not Completely Capture the 4 Corners.

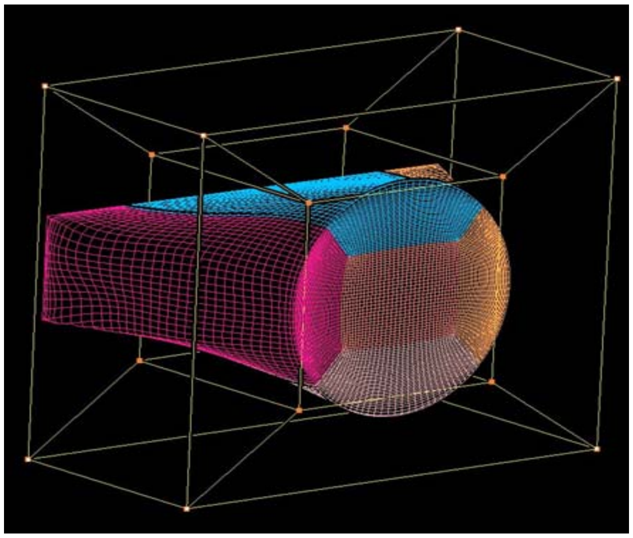


Figure 13. Topology and Grid for Baseline Shape Transition Case

The topology is defined to be general enough to handle the changes as the shape transition surface is updated, and generate an appropriate grid. The grid has five zones. Each zone is broken into 48x48x48 cells for a total of 552,960 cells. The simulations were run frictionless, so no particular wall clustering was specified.

Baseline CFD Results

In addition to the decision to assume inviscid flow, the simulation was run assuming perfect gas. In the actual design process, as the design is refined and the design space narrows, more detail can be added to the models as required.

A uniform inflow profile was used, as was an extrapolate outflow boundary condition. Table 2 lists the aero-thermal values specified at the inflow boundary.

Table 2. Uniform Inflow Boundary Condition Specification

Mach Number	Temperature	Pressure	Gamma	Gas Constant
---	K	Pa	---	J / (kg K)
2.5	300	50000	1.4	287

The inflow boundary conditions were chosen in part to ensure supersonic flow was maintained at the outflow boundary so that the boundary condition specified remained well-posed. The inflow velocity vector was aligned with the x-direction.

The wall pressure profile for the top and side of the duct are shown in Figure's 14 and 17. The shape of the compression wave is evident in Figure 14, the top view. The waves generated by the side wall compression coalesce just upstream of the exit plane. The profile at the exit plane is indicative of a wave pattern that would continue to reflect down a duct.

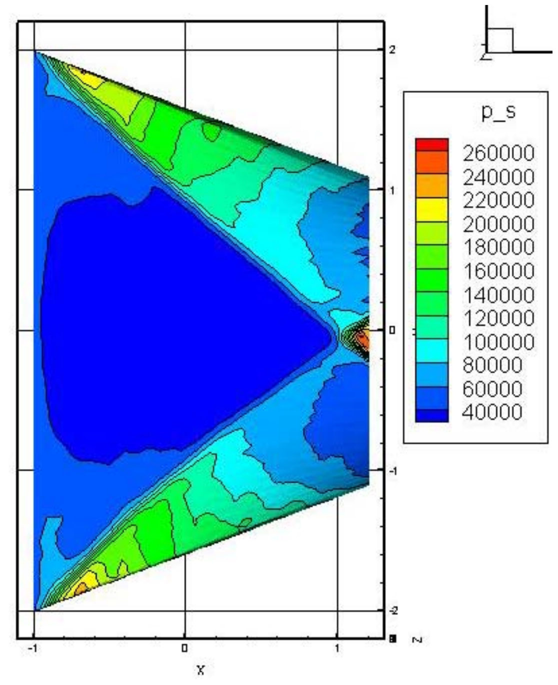


Figure 14, Uniform Inflow Wall Pressure Contour Plot – Baseline

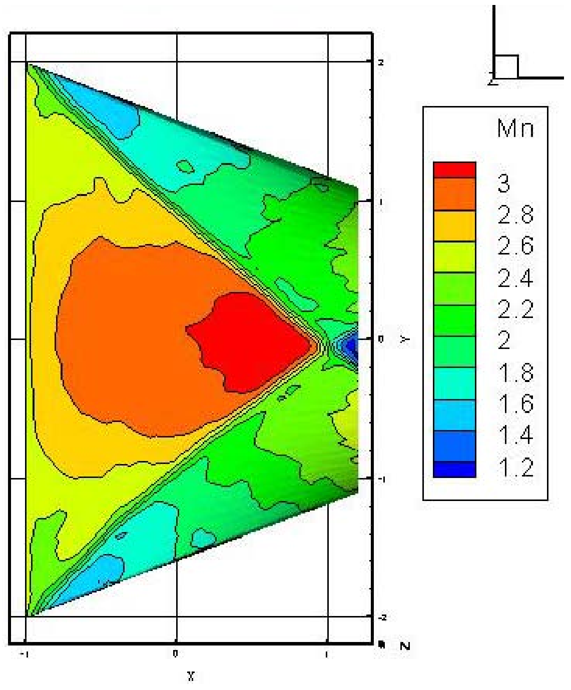


Figure 15, Uniform Inflow Mach Number Contour Plot - Baseline

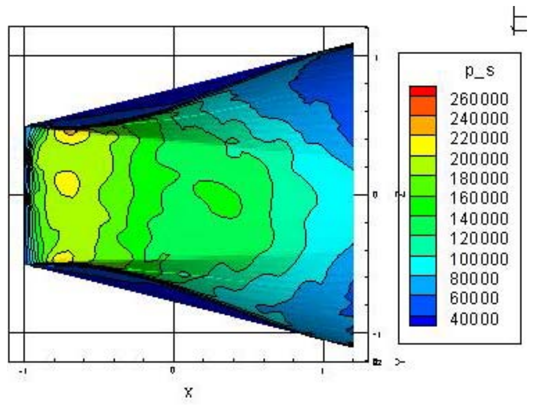


Figure 16, Uniform Inflow Side-Wall Pressure Contour Plot - Baseline

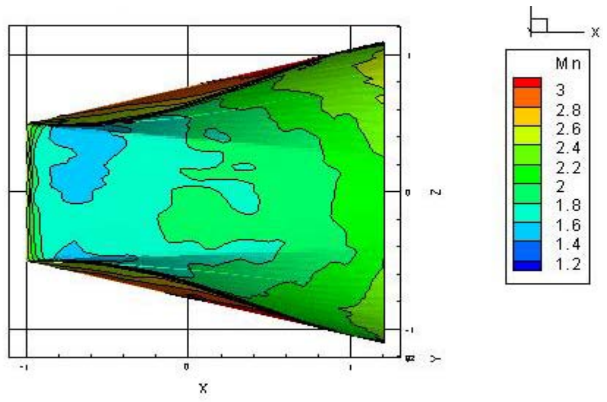


Figure 17, Uniform Inflow Mach Number Contour Plot - Baseline

The total pressure loss for this case was calculated to be 40.32%. Obviously this is a large number and no real system could tolerate such a loss in total pressure and stay operational. This is, however, an excellent candidate for optimization!

Surface Generation

A utility called *subdivision* was used to generate the surfaces for the analysis. *Subdivision* allows the user to define a set of control points through which the surface passes. Additionally, the user can control the normal to the surface at those points. The user communicates with the utility via an input file.

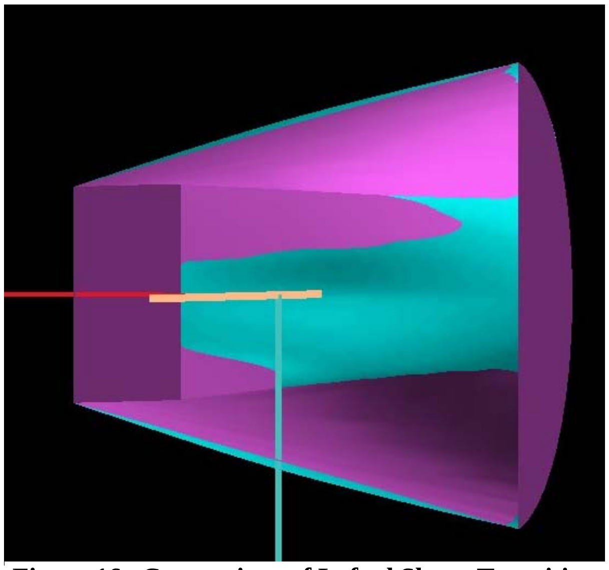


Figure 16. Comparison of Lofted Shape Transition to Transition Generated from the *Subdivision* Surface Generation Utility

A comparison of the baseline lofted surface and the surface generated by *subdivision* is shown in Figure 16. The *subdivision* surface was generated by defining two rings of eight control points at the entrance (the rectangular section) and exit (the round section) of the shape transition. These points were then input into *subdivision* which generated a surface passing through the points. These control points remained fixed during the optimization process. Once the *subdivision* surface was generated, a third set of eight control points was defined mid-span. Each of these control points are on the surface and between the corresponding control points on the inflow and outflow boundaries. This is shown graphically in Figure 17. Since these points were selected graphically, they do not lie exactly on the surface, nor are they necessarily symmetric. This will not affect the demonstration of the tool and should not affect the quality of the answer.

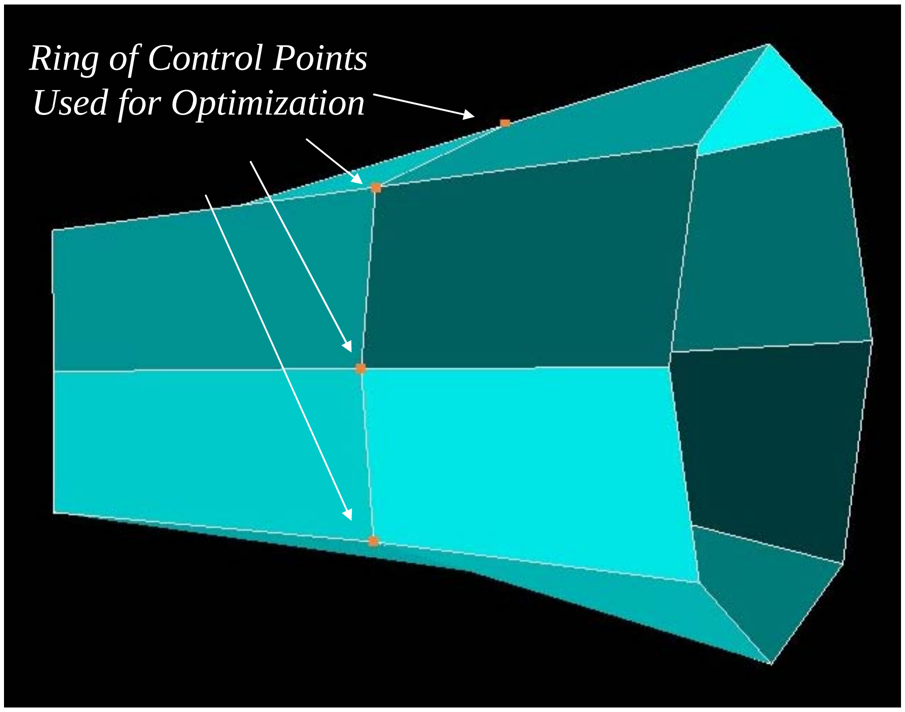


Figure 17. Surface with Control Points to Perform Optimization

Optimization Set-Up

In order to set up the optimization, it is necessary to have a good understanding of two things, the inputs to the analytical model with at least a basic understanding of how the inputs will affect the results and how to process the results of the analysis to arrive at an objective function that is to be optimized.

As described earlier, the inputs to *subdivision* are the control points that the surface passes through and the normal to the surface at those points. Each of the control points in Figure 17 has four degrees of freedom (DOF), three spatial and the surface normal at

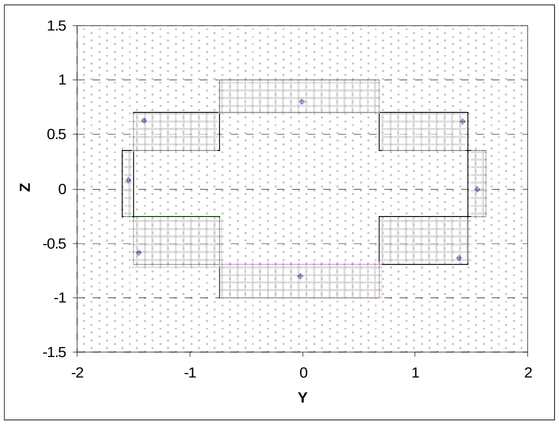


Figure 18. Mid-Span Control Points and Their Associated Range of Motion

each point. For a ring of eight control points, this results in potentially 32 independent variables to be optimized. This is a large number of combinations to be investigated. Initially, the optimization will be limited to 16 DOF's, two for each mid-span control point. The control points will be constrained to stay mid-span and the normal to each point will not be set a-priori. The initial conditions, along with the bounds for each point, are shown in Figure 18.

Note in the figure that the control points at the Z ~= 0 line have the greatest allowable motion, followed by the "corner" points. The Y ~= 0 points have the tightest control on their movement. This is an artifact of the way the mid-span control points were chosen (inline with the control points located on the inflow and outflow boundaries) along with the desire not to allow multiple points-of-inflection between any three control points. In practice, this requires the bottom control point always be below the adjacent "corner" points. Like wise, the top and side points should always be out-board of their adjacent "corner" points. In future set-ups, the design space can be more open with point-of-inflection type requirements accomplished with constraints.

The second part of setting up the optimization case was processing a figure of merit (FOM) to be either minimized or maximized. The FOM chosen for this work is to minimize the total pressure loss across the shape transition. Recall that the area of the shape transition section decreases five percent causing the supersonic flow to compress. There is an inherent loss in total pressure across compression waves so an obvious choice in performing the optimization is to minimize the total pressure loss across the shape transition. In order to ensure the FOM has stabilized, a mass averaged total pressure loss calculation was added to the VULCAN output. As described earlier, the FOM can be calculated a variety of different ways and does not require modification of the source code of the solver. This does, however demonstrate the significant utility of having the source code available.

The optimization routine used comes from the SGOPT (Stochastic Global OPTimization) library, which is included with the base DAKOTA install. This is a non-gradient based library of methods that include both local (Solis-Wets and pattern search) and global (genetic patter search, evolutionary pattern search and stratified Monte Carlo) algorithms. For this work, the pattern search scheme was chosen and run over 50 iterations.

Optimization Results

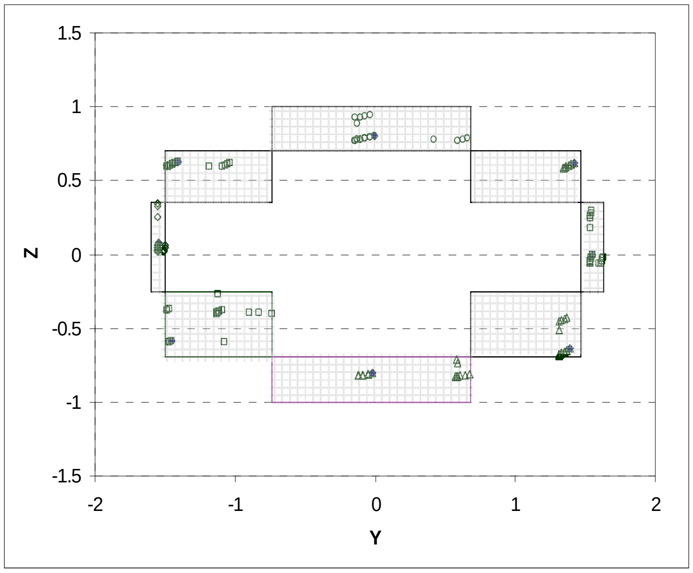


Figure 19. Scatter Plot of Control Point Location for the 50 Optimization Iterations

Over three days, there were 50 geometries analyzed, with 50 different grids, without human intervention. New surfaces were generated based on the ring of control points whose positions were determined by the optimizer. A plot of the control point locations for each of the iterations is shown in Figure 19. The locations of some of the control points did not vary much at all while others were more distributed about the design space. This is evidence that 50 iterations is not enough to sample the design space of 16 independent variables.

Even though the analysis has not thoroughly probed the design space, it is still possible to see if the there has been any improvement in total pressure loss. Figure 20 shows the percent decrease in total pressure loss on an iteration-by-iteration basis. As can be seen, there are many trials where there is significant improvement of the lofted surface. Sixteen of the 50 runs had reduced the total pressure loss by at least 5 %. The best design generated to date is iteration number 20 which has a total pressure loss of 37.2 %, a 7.7% reduction in total pressure loss.

Surprisingly, only three of the simulations showed an increase in total pressure loss. These runs, combined with the three runs that showed less than 0.5% improvement, accounted for only 12 % of the iterations. This is due to in part to the pattern search scheme looking for a local, verses a global, optimum. Overall, a 7.7% reduction in total pressure loss is a promising result for so few iterations.

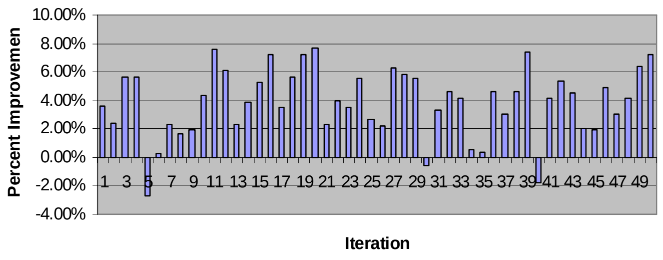


Figure 20. Percent Decrease in Total Pressure Loss on an Iteration By Iteration Basis

The comparison of the iteration 20 surface and the baseline surface is shown in Figure 21. It is important to note that flow in this picture is from right-to-left. The aqua surface is the baseline and the blue surface is the surface from iteration number 20. Note that the baseline surface is not symmetric. The lower surface is pulled up into the flow path relative to the baseline and the upper surface is slightly above the baseline.

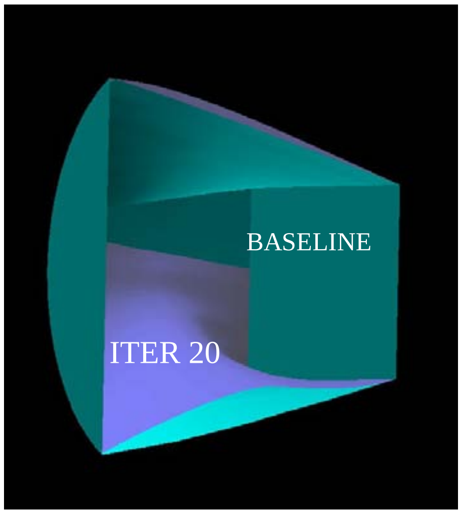


Figure 21. Comparison of Baseline Lofted Surface and the Iteration 20 "Best" Surface Generated by the Optimizer

If the optimization case were allowed to continue to run, it is believed that, for this case, a symmetric solution would result. Unfortunately, time and computational constraints prohibited this from being done. The reason that symmetry was not exploited is found in the follow on work which used inflow profiles that were not necessarily symmetric. This work is briefly described in the following section.

The simulation results from twentieth iteration are shown below. In Figure 22 you can see the asymmetry in the wall pressure distribution viewed from the top.

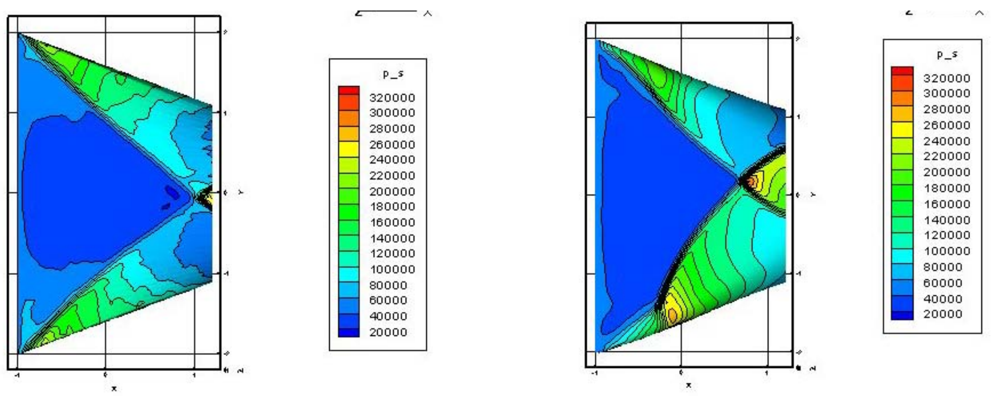


Figure 22. Top Wall Pressure Profiles for the Baseline and Iteration-Twenty Cases

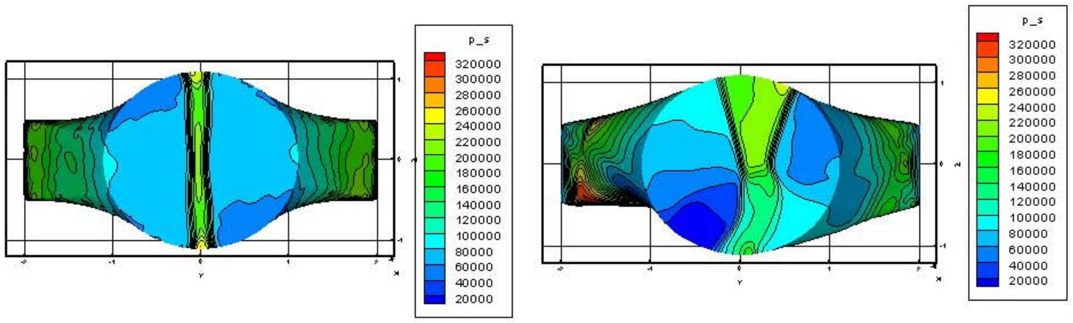


Figure 23. Exit Plane Pressure Contours for the Baseline Case and Iteration-Twenty Cases

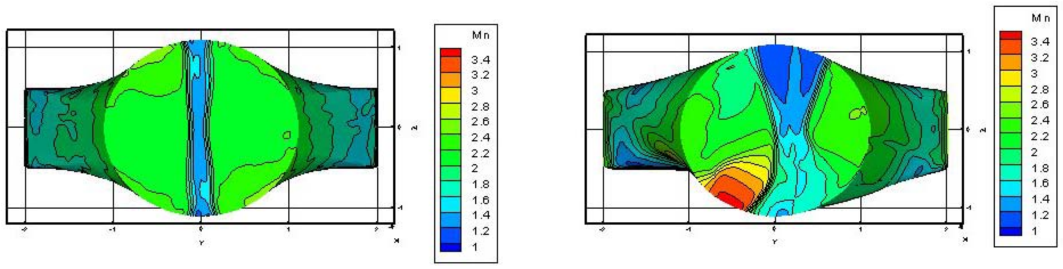


Figure 24. Comparison of Outflow Mach Number for the Baseline and Iteration-Twenty Cases

The flow exiting the shape transition from the optimizer is much more distorted than the baseline case. Additionally, the minimum Mach number at the exit plane is lower in case twenty (Mn_{MIN} =1.05) than the baseline simulation (Mn_{MIN} = 1.2). This causes some concern that, although the residual dropped four orders of magnitude and was well behaved, the exit boundary condition may not be well posed for all of the solutions and should be checked.

The convergence history, conservation of mass and total pressure loss parameters are all shown in Figure 25. The percent error in mass flow is driven to zero and the total pressure loss calculation is steady. Therefore, even though the residual is still dropping, the FOM is steady and the solution is converged enough to be useful.

When assessing the solution it is important to keep in mind that the simulation is frictionless and the flow can turn along wall angles that are unrealistic. This behavior will allow for geometries that would normally separate to essentially not be penalized. If this were for a design to be built, the simulations would be run viscous.

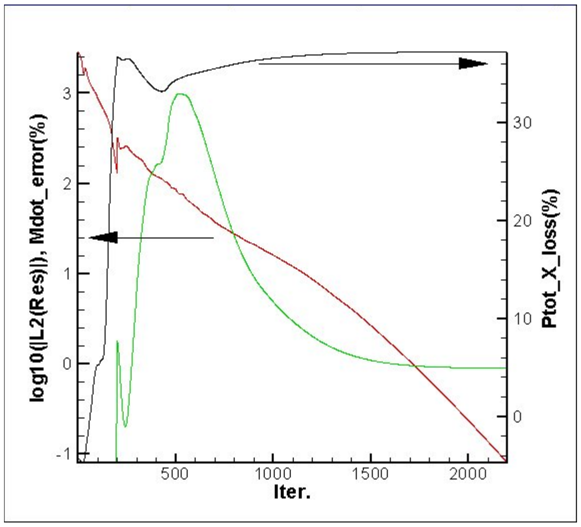


Figure 25. Clean Case 20 Convergence History

Four Inflow Profiles

An additional benefit of using the OH-grid methodology was that it provided the ability to rapidly set different inflow profiles. There were four inflow profiles used when investigating the shape transition optimization scheme. These profiles include a clean (NO DIST) inflow profile and three distorted profiles; circumferential (CIRC), Outer Diameter (OD) and Inner Diameter (ID). For the purposes of this work, distortion is defined as a mismatch in total pressure. OD distortion is defined to be a total pressure deficit near the wall or the OD of the duct while ID distortion is lower total pressure near the core or the ID of the duct. CIRC distortion is defined to be a mismatch in total pressure from one side of the duct to the other.

To create the distortion, the velocity was increased and decreased by 10% and the static density was altered by 10% in the opposite direction of the velocity change. The static temperature was held constant. This approach conserved mass and momentum but created a mismatch in velocity, static and total pressure. The value of each of these quantities is shown below in Table 3.

Table 3, Thermodynamic Inputs for Clean and Distorted Inflow USed During the Shape Transition Demonstration

	Static density	Axial velocity	Static Temperature	Total Pressure
Low Velocity Flow	0.38327526	1008.3	500.0	635972
Nominal Flow	0.348432	1120.5	500.0	854297
Hi Velocity Flow	0.31358885	1232.7	500.0	1131316

Baseline Distortion Cases

The Mach number profiles for each of the inflows are shown below in Figure 26. The solutions were generated using the baseline lofted surface. It is interesting to see how the distorted profiles influence the shock structure. The OD distortion pulls the pressure wave intersection back in the duct while the ID distortion pushes the intersection forward in the duct. The circumferential distortion causes the pressure waves to intersect in the side of the duct with the higher initial velocity.

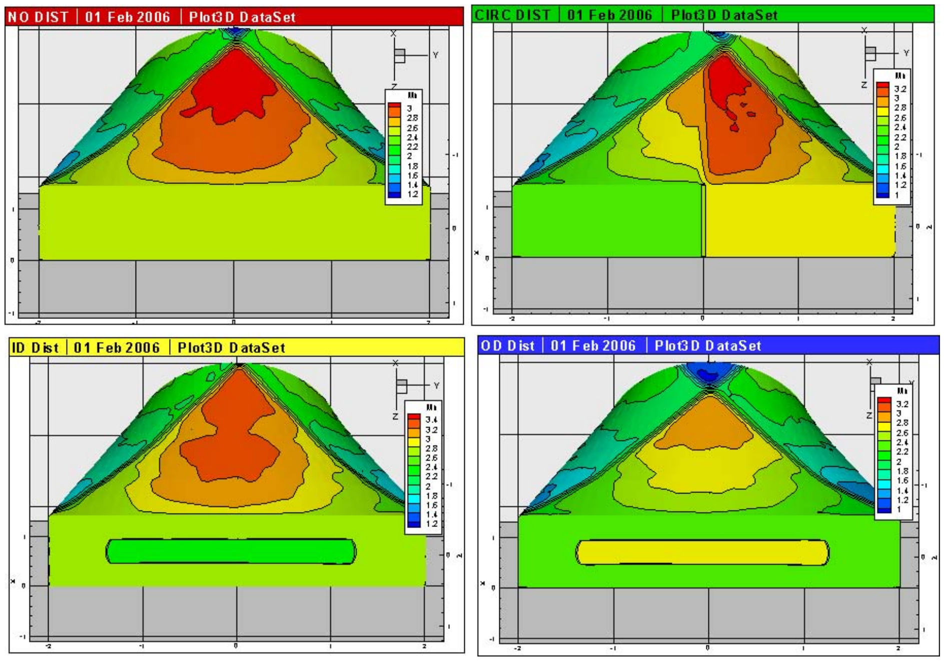


Figure 26. Four Inflow Mach Number Profiles Used for Shape Transition

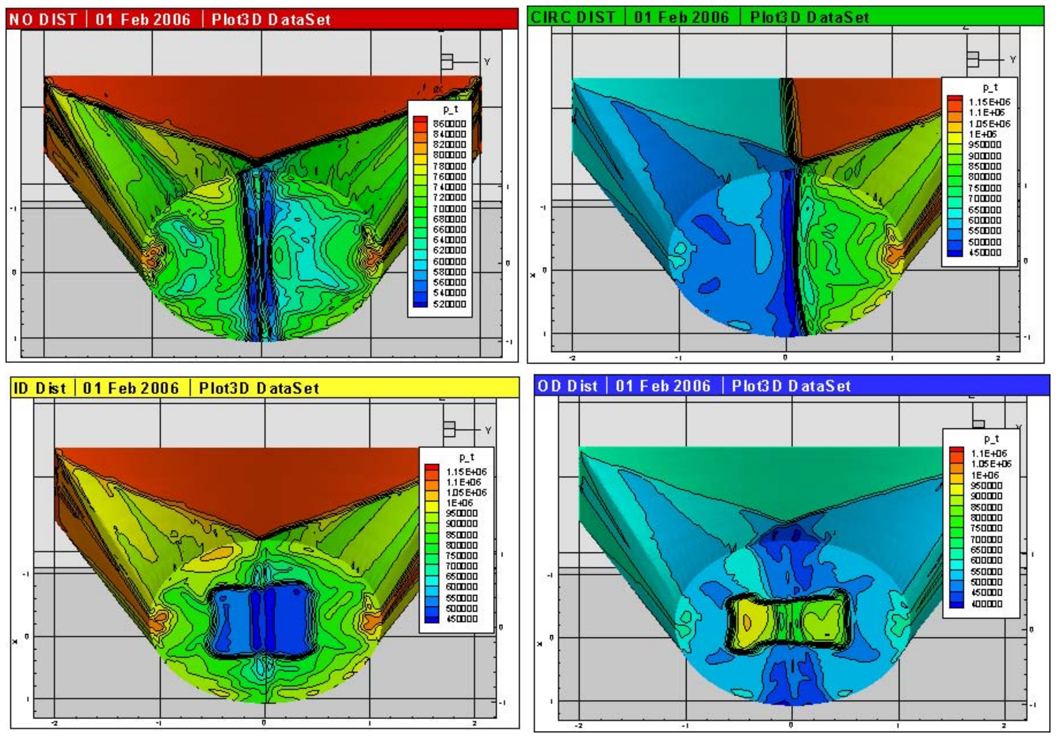
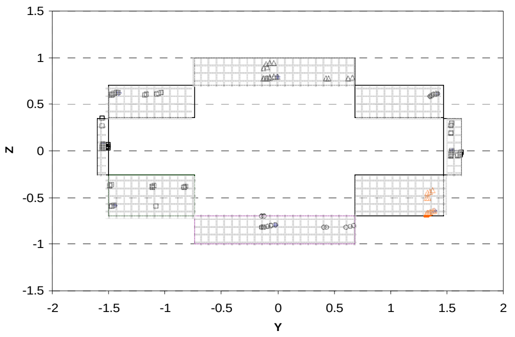


Figure 27, Total Pressure Profiles at the Exit of the Baseline Shape Transition

The total pressure pattern at the exit of the duct for each distorted inflow profile is shown in Figure 27. The duct with no distortion has the most uniform total pressure profile at the exit. The ID, OD and CIRC cases all still have high levels of distortion at the exit plane. On a side note, it would be very interesting to add the isolator section to the analysis and back pressure the system to see how much the inflow distortion affects the isolator's capability.

OD Distortion

The OD distortion case started similarly to the no distortion case. This was expected since the optimization method used was not gradient based and the number of independent variables was large. It is, however, counter intuitive that out of 49 trials, only one would have greater total pressure loss than the baseline. This seems to imply that a lofted surface is a very poor method of generating a shape transition for a supersonic duct.



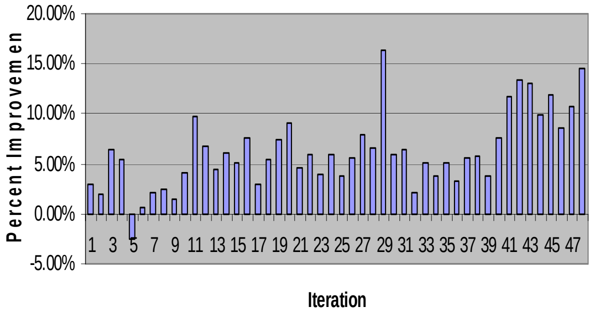


Figure 28, Control Point Locations and Resulting Total Pressure Loss for Each Iteration

Run 29 showed the greatest total pressure benefit, this simulation is detailed below.

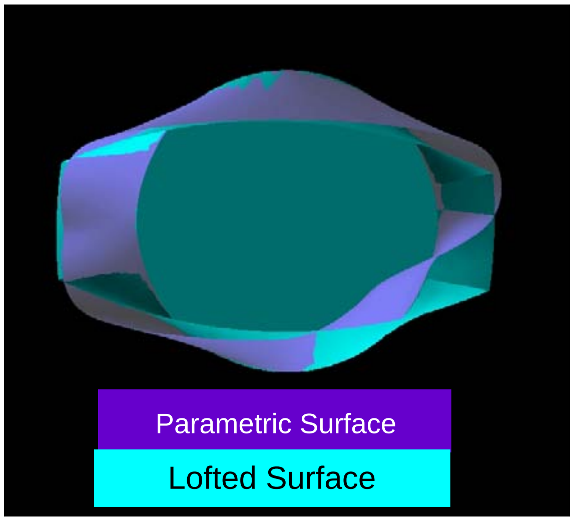


Figure 29, Comparison of Baseline Surface and the Surface of Iteration 29.

The shape transition surface, compared to the lofted surface, is shown in Figure 29. Since the surface is not symmetric, it is probably not a design that would be considered. In fact, when using this tool to develop components for manufacture, manufacturing rules such as maximum/minimum corner radii and/of maximum change in surface and desired symmetry will be input via problem set-up and/or constraints in the optimization process.

As can be seen in Figure 30, the CFD simulation was run for 2700 iterations. The residual dropped approximately five orders of magnitude and the mass flow error was steady at just about zero percent error. The total pressure loss calculation was also steady for the last 400-500 iterations

Figure 31 shows the total pressure, static density and Mach number profiles for the OD distortion case. Note that the Mach number profile at the exit plane is greater than Mach 1.2 everywhere, ensuring the exit boundary condition is well posed.

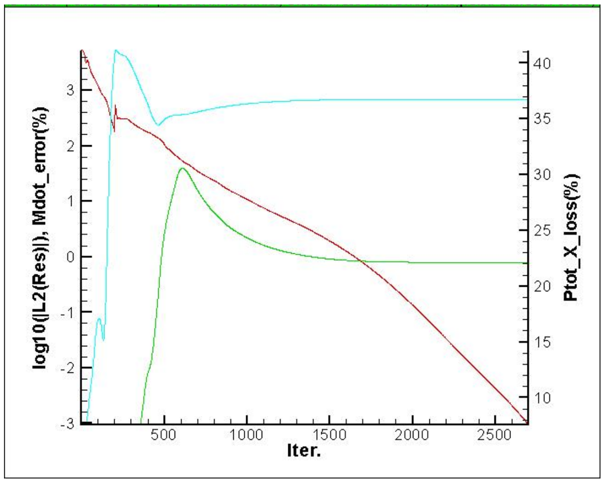


Figure 30, Convergence and FOM History for Iteration 29 of the Optimization Using OD Distortion

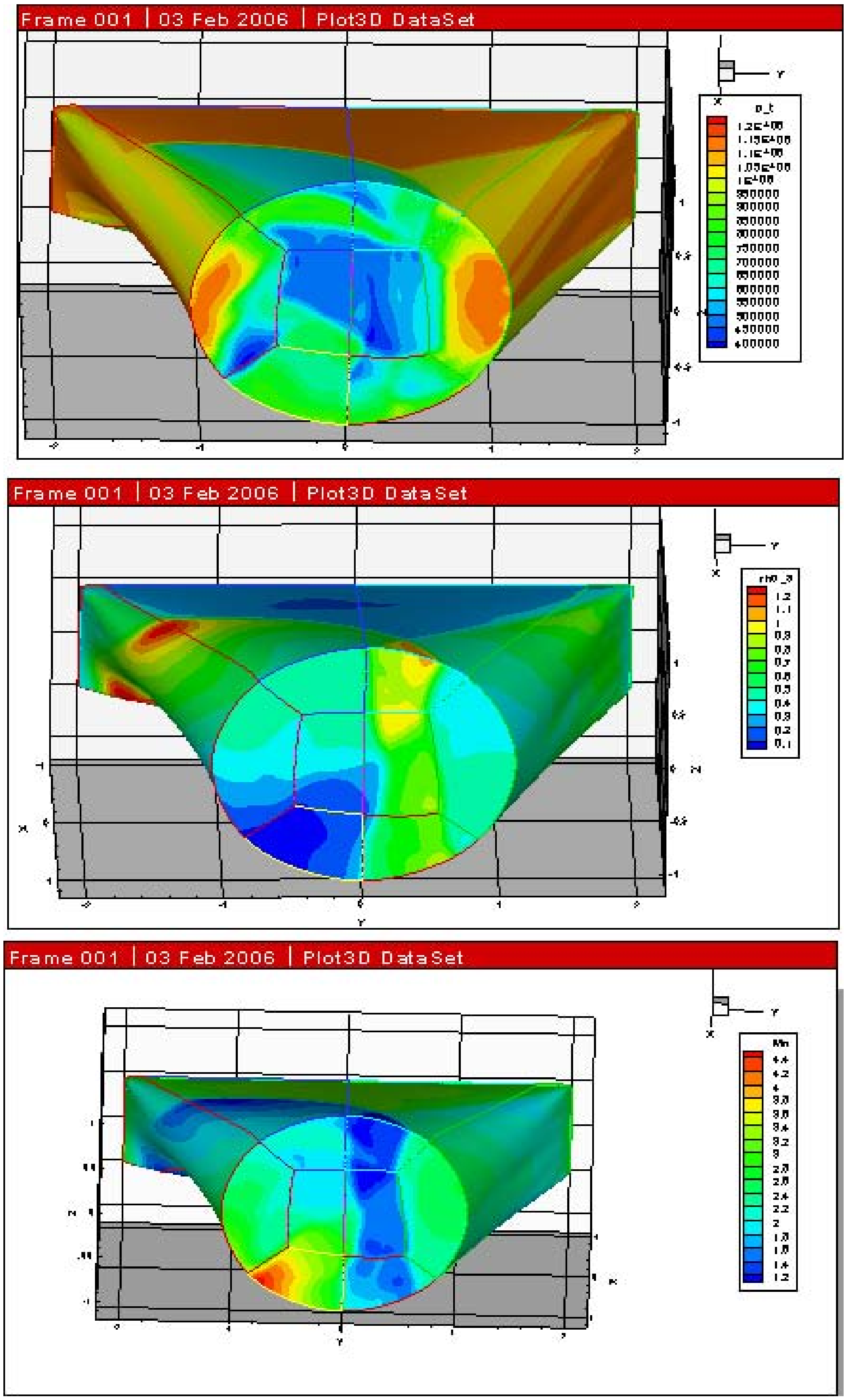


Figure 31, Total Pressure, Static Density and Mach Number Profiles for Iteration 29 of the OD Distortion Optimization

ID Distortion

The ID distortion case also started similarly to the no distortion case. The ID case was run for 55 iterations and showed a large improvement of over 15%.

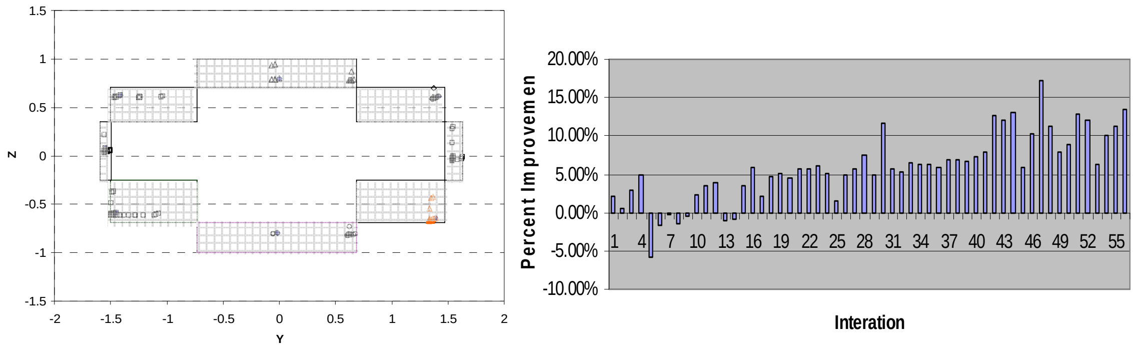


Figure 32, Control Point Location and Relative Total Pressure Loss of the ID Distortion Optimization

Upon inspection of the solutions, many of the best performers had a separated flow region at the exit plane, which is not a valid date point. Evaluating each iteration in order of highest to lowest FOM, revealed the best performer that had a valid solution was iteration number 28, a which had 7.5% gain in total pressure. The surface for this iteration is shown in Figure 33. The convergence history of the simulation is shown in Figure 34.

As with the other cases, much further refinement is required for a design-quality solution

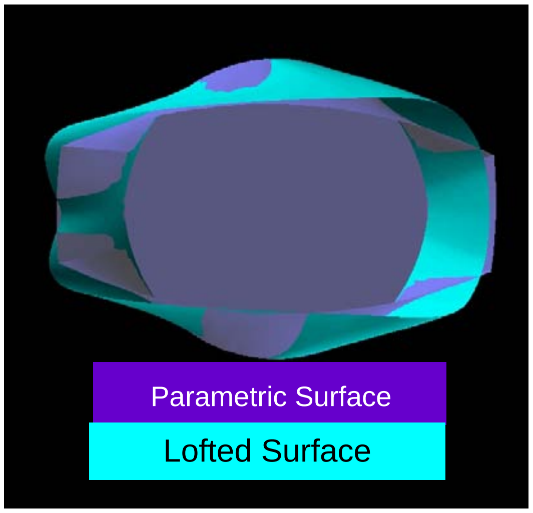


Figure 33, Comparison of Lofted and Optimized Surface for the ID Distortion Case

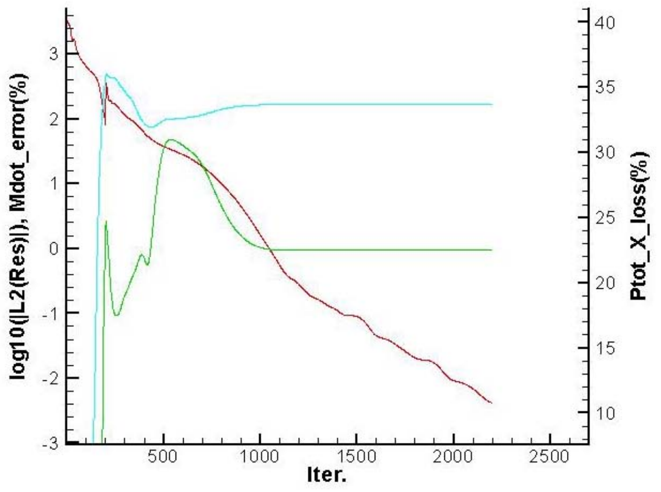
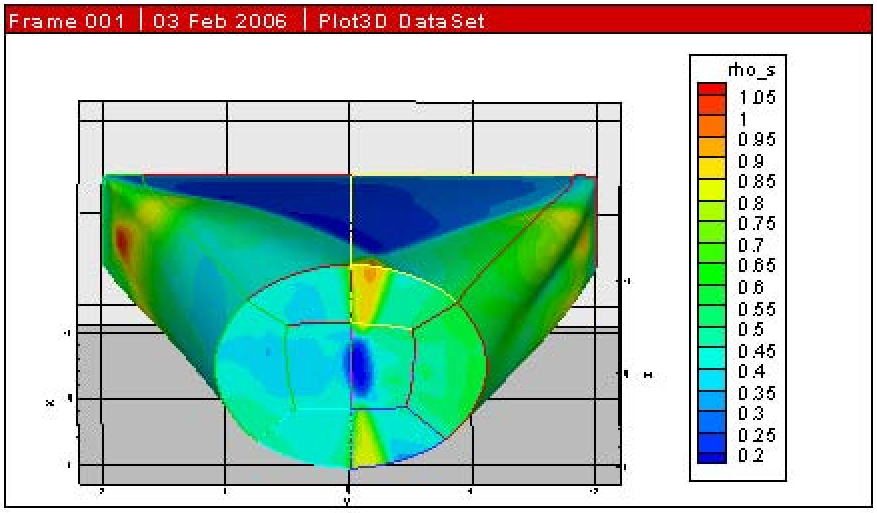
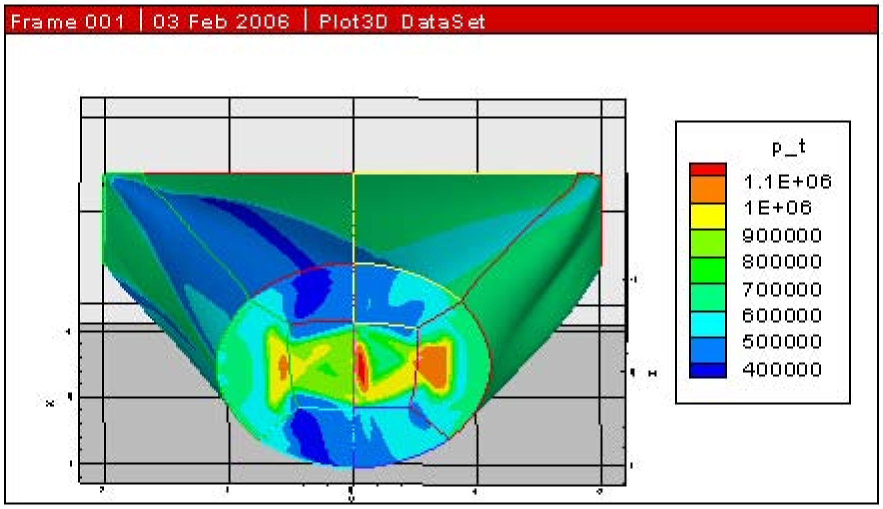


Figure 34, Convergence History for Iteration 28 Analysis

Details of the analysis are shown in Figure 35 below. These show the total pressure, static density and Mach number profiles for iteration 28.





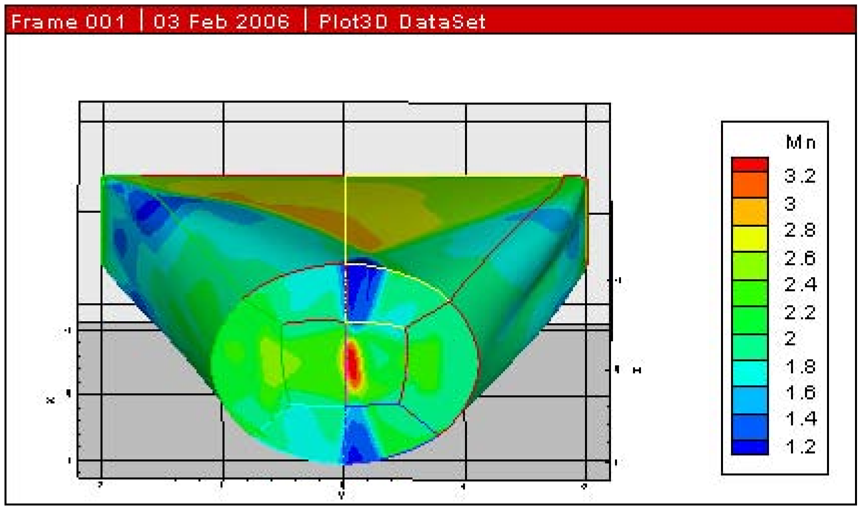


Figure 35, Total Pressure, Static Density and Mach Number Profiles for Iteration 28 of the ID Distortion Optimization

Circumferential Distortion

Unfortunately, time ran out before results for the circumferential optimization analysis were available.

CONCLUSION / LESSONS LEARNED

An approach to an integrated design optimization and engineering analysis tool has been demonstrated. This tool leverages the significant work required to set-up a detailed numerical analysis. The additional work required to execute the optimization includes formally defining the FOMs, setting up the DAKOTA input file and possibly re-defining the geometry and topology for parametric grid generation. Setting up the DAKOTA input file is a straight forward task. Since the process requires a formal definition of the FOMs, the system documents itself and is very repeatable. Including grid generation in the loop requires some additional effort, however, the potential gains in component performance and system operability are significant.

An additional benefit of the system is that it can be used to automate performance map generation. The same iterator that is required by the optimizer, in this case DAKOTA, can be used to create a performance map with little additional work. The engineer is relieved of the repetitive tasks of input file creation, job submission, etc. Rather, he can concentrate on the product, not the process.

Optimization Techniques

Of the two main types of optimization schemes, the gradient based methods were more robust because they only make small changes to the geometry, which is easier for the topology to handle. The two main drawbacks to running these types of schemes are that sometime large initial perturbations must be made to avoid the numerical noise and the schemes find a local, rather than a global optimum.

The non-gradient based schemes, designed to find a global solution, sample the entire design space. Some do so more aggressively than others and require a more robust topology be defined.

The preferred method of zeroing in on the optimum design depends on the starting point and design space. If it is early in the design and the design space is large, it may be prudent to start with a lower fidelity model and sample the entire design space with a sparse number of iterations. Once the non-gradient based optimization is in the ball park, switch over to a gradient based technique and run with a higher fidelity model.

Lots of Data

The main hurdle to integrating this tool into the industrial design process is in handling the enormous amounts of data generated. For our shape transition case where every effort was made to keep the simulation size down, generated over 80 gigabytes of data! An automated method of checking solutions for convergence and accuracy is required.

This could mean generating specialized scripts to interact with individual packages, or potentially to use the CGNS standard as a more general means of communication.

Finally, some manner of ensuring that the simulation is feeding the optimizer valid data is required. This can be done a variety of different ways and no one way will be the answer for all cases. For instance, one case may require restricting the computation domain, another may be done completely by post processing the results while yet another way may involve tweaking the analytical code itself. This is where the expertise of the users will be required as they apply the tool to their specific area of interest.

APPENDICES

Appendix 1.A, DAKOTA Input File for Fuel Injector Case

```
# 1st run with DAKOTA and Vulcan
# on the cluster ...
# gradient-based unconstrained optimization


variables,                                        \
       continuous_design = 3                     \
         cdv_descriptor       'u1'  'w1'  'exp1'      \
         cdv_initial_point    20     20     2         \
         cdv_lower_bounds     10    10     2          \
         cdv_upper_bounds     40    40     10

interface,                                        \
       application system,                        \
         asynchronous                            \
         evaluation_concurrency = 1              \
         analysis_driver =     'vulcan_driver' \
         parameters_file =     'params.in'     \
         results_file =        'results.out'   \
         file_tag                                \
         file_save

responses,                                        \
       num_objective_functions = 1               \
       no_gradients                               \
       no_hessians

method,                                           \
#       dot_bfgs                                  \
       sgopt_pattern_search                       \
       solution_accuracy = 1.0e-4                \
       initial_delta = 0.5                       \
       threshold_delta = .0001                  \
       max_function_evaluations = 50            \
       exploratory_moves best_all               \
       contraction_factor = 0.75                \
       output quiet

strategy,                                        \
       single_method                             \
#       graphics                                 \
       tabular_graphics_data
```

Appendix 1.B, VULCAN Input File

```
'$*****************************************************************$' 0.0
'$*****************************************************************$' 0.0
'$****  Shaped Transition Duct                                  ***$' 0.0
'$*****************************************************************$' 0.0
'$*****************************************************************$' 0.0
'$**************** Begining of general control data ******************$' 0.0
'$---------------- Parallel processing control data ------------------$' 0.0
'PROCESSORS'          8.0     (No. of cpus to use)
'MESSAGE MODE'        1.0     (Message passing strategy: 0=stnd., 1=buffered)
'LOAD BALANCE MODE'   0.0     (Load balancing algorithm: 0=stnd., 1=pmetis, 2=kmetis)
'$---------------- Geometric model type ------------------------------$' 0.0
'THREED'           1.0   (twod, axisym, threed)
'$---------------- Grid file data ------------------------------------$' 0.0
'GRID FORMAT'      3.0   (1=s.b.form, 2=s.b.unform., 3=m.b.form., 4=m.b. unform)
'GRID'             0.0   (0=plot3d->3d ; plot2d->2d/axi, 1=plot3d->all)
'baseline_8blk.p3d'
'GRID SCALING FACTOR' 0.0254  (Converts grid units to meters)
'$---------------- Restart file data ---------------------------------$' 0.0
'RESTART OUT'      0.0
'Restart_files/baseline_8blk_rlx3d.restart'
'RESTART OUT INTERVAL'  100.0
'$------------------- Output control data ----------------------------$' 0.0
'WARNING MESSAGES'     0.0   (0=None, 1=wall funct., 2=temp. limit, 3=dq limit, 4=all )
'PLOT ON'              4.0   (1=s.b.frm., 2=s.b.unfrm., 3=m.b.frm., 4=m.b.unfrm.)
'32 BIT BINARY'        0.0   (Write 32 bit unformatted PLOT3D files)
'PLOT NODES'           0.0   (Create PLOT3D files using data averaged to the nodes)
'PLOT FUNCTION'        6.0   (Create PLOT3D function file containing variables below)
 'DENSITY', 'VELOCITY', 'PRESSURE', 'TEMPERATURE', 'MACH NO.', 'EDDY VIS. RATIO'
'$---------------- Gas thermo, diffusion, and reaction model data -----$' 0.0
'GAS/THERMO MODEL'     0.0   (0=const. gamma, 1=mix therm perfect, 2=n/a)
'GLOBAL VISCOUS'       0.0   (Solve Navier-Stokes equation using global algor.)
'VISCOSITY MODEL'      1.0   (n/a=power law, 1=Sutherlands law)
'$---------------- Free stream gas angle data ------------------------$' 0.0
'ANGLE REF. FRAME'     0.0   (0=ALPHA in xy plane, 1=ALPHA in xz plane)
'ALPHA'                0.0   (Angle of attack measured C.C.W in degrees)
'BETA'                 0.0   (Angle of yaw measured C.C.W in degrees)
'$---------------- Force and Moment integration control data ----------$' 0.0
'INTEGRATION REF. LENGTH'    1.0   (reference length for inegrated force and moment
coeff's)
'INTEGRATION REF. AREA'      1.0   (reference area for inegrated force and moment
coeff's)
'INTEGRATION X SCALE FACTOR' 1.0   (X component scale factor for inegrated force and
moment coeff's)
'INTEGRATION Y SCALE FACTOR' 1.0   (Y component scale factor for inegrated force and
moment coeff's)
'INTEGRATION Z SCALE FACTOR' 1.0   (Z component scale factor for inegrated force and
moment coeff's)
'MOMENT REF. X'              0.0   (X coordinate of datum for moment integration)
'MOMENT REF. Y'              0.0   (Y coordinate of datum for moment integration)
'MOMENT REF. Z'              0.0   (Z coordinate of datum for moment integration)
'INTEGRATION REF. PRESSURE' -1.0   (Integration ref. pressure, Pascals)
'INCLUDE SLIP WALLS'         0.0   (Include all slip wall b.c.s in the integration)
'$---------------- Figures of Merit computation control data ----------$' 0.0
'COMPUTE TOTAL PRESSURE LOSS' 1.0   (Compute the total pressure loss)
'$---------------- Reference condtion data ---------------------------$' 0.0
'NONDIM'               1.0   (0=non.dimen., 1=dimen. static, 2=dimen. total)
'MACH NO.'             2.5
'GAMMA'                1.4
'STATIC TEMP.'         300.0
'GAS CONSTANT'         287.0
'STATIC PRESS.'        50000.0
'UNIT REYNOLDS NO.'   -1.0
'REFERENCE LENGTH'     1.0
'SUTHERLANDS LAW S0'   110.555556
'SUTHERLANDS LAW T0'   273.111111
'SUTHERLANDS LAW MU0'  0.1716E-04
'LAM. PRANDTL NO.'     0.72
'TURB. PRANDTL NO.'    0.90
'$---------------- Turbulence model data ------------------------------$' 0.0
'TURB. MODEL'     0.0
  'K-OMEGA' (LAMINAR, K-EPSILON, K-OMEGA, LOW RE K-OMEGA, MENTER, MENTER-SST)
  'TURB. INTENSITY'    0.01
  'TURB. VISC. RATIO'  0.10
  'BOUSSINESQ REY. STRESS'     0.0
  'DURBIN REALIZABILITY'       1.0
  'NO 2/3 RHOK IN REY. STRESS' 0.0
  'INITIAL BOUNDARY LAYER THICKNESS' 0.010
```

```
'$------------------- Runge-Kutta scheme coefficients -----------------$' 0.0
'NSTAGE'           3.0   (no. of Runge-Kutta Stages)
 0.333333333333, 0.5, 1.0
'$------------------- Boundary and cut control -----------------------$' 0.0
'FLOWBCS'          22.0   (no. of boundary conditions to be specified)
'CUTBCS'              13.0   (no. of C(0) cut conectivity conditions)
'PATCHBCS'           0.0   (no. of Non-C(0) cut conectivity conditions)
'BLOCKS'           8.0   (no. of blocks)
'BLOCK CONFIG.'    1.0    (no. of lines of block configurations input)
'BLK I-VISCS. J-VISCS. K-VISCS. TURBULENCE PLOT SOLVER REGION'
 0     'N'      'N'       'N'       'N'      'Y'  'E/A'   1
'REGION CONFIG.'  1.0  (no. of regions the blocks are grouped into)
'$****  Region 1   control input ************************************$' 0.0
'LDFSS   KAPPA     LIMITER   LIM.-COEF.      ENTRP(U)        ENTRP(U+a)'
      3, 3, 3,  2, 2, 2,  2.0, 2.0, 2.0,  0.0, 0.0, 0.0,  0.0, 0.0, 0.0
'FMGLVLS NITSCG1 NITSFG  #1ST-ORD.-C.G./ITER. RES.;REL.,ABS.'
   2,    100,   500,         -2,           -99.0, -99.0
'M.G.-CYCLE #C.-G. DQ-SMOOTH DQ-CORR. DAMP-MEAN DAMP-TURB.'
   'I',      1,     0.10,     -0.25,    0.50,    0.25
'TURB. CONVEC.-ORDER DT-RATIO NON-EQUIL. POINT-IMP. COMP.MODEL C.G.WALL-B.C.'
    '2ND',          1.0,     25.0,       'Y',       'Y',       'WMF'
'SCHEME TIME-STEP IT-STATS CFLMIN ADPCFL #CFL-VAL VISC-DT IMP-BC REG-REST.'
 'R3D',  'LOCAL',   10,     0.5,   'Y',     4,       'Y',    'Y',   'Y'
'SWEEP-DIR NSTART NMOD KITER'
   0,        1,    0,   2
      1,     100,     101,      200
   0.10,   400.0,    0.10,    400.0
'!****************** End of general control data ********************!' 0.0
'BC  NAME    BLK  FACE   PLACE DIREC1 BEGIN   END  DIREC2 BEGIN   END  IN-ORDER  BC TYPE'
'SUP-OUT.'    1   'K'  'MIN'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'EXTRAP'
'ADB-WALL'    1   'I'  'MAX'  'J'   'MIN'  'MAX'  'K'   'MIN'  'MAX'    0     'SWALL'
'SUP-INF.'    1   'K'  'MAX'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'AREFFIX'
'ADB-WALL'    2   'I'  'MIN'  'J'   'MIN'  'MAX'  'K'   'MIN'  'MAX'    0     'SWALL'
'SUP-OUT.'    2   'K'  'MIN'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'EXTRAP'
'SUP-INF.'    2   'K'  'MAX'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'AREFFIX'
'ADB-WALL'    3   'I'  'MIN'  'J'   'MIN'  'MAX'  'K'   'MIN'  'MAX'    0     'SWALL'
'SUP-OUT.'    3   'K'  'MIN'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'EXTRAP'
'SUP-INF.'    3   'K'  'MAX'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'AREFFIX'
'SUP-OUT.'    4   'K'  'MIN'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'EXTRAP'
'ADB-WALL'    4   'I'  'MAX'  'J'   'MIN'  'MAX'  'K'   'MIN'  'MAX'    0     'SWALL'
'SUP-INF.'    4   'K'  'MAX'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'AREFFIX'
'SUP-OUT.'    5   'K'  'MIN'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'EXTRAP'
'ADB-WALL'    5   'I'  'MAX'  'J'   'MIN'  'MAX'  'K'   'MIN'  'MAX'    0     'SWALL'
'SUP-INF.'    5   'K'  'MAX'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'AREFFIX'
'SUP-OUT.'    6   'K'  'MIN'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'EXTRAP'
'SUP-INF.'    6   'K'  'MAX'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'AREFFIX'
'ADB-WALL'    7   'I'  'MIN'  'J'   'MIN'  'MAX'  'K'   'MIN'  'MAX'    0     'SWALL'
'SUP-OUT.'    7   'K'  'MIN'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'EXTRAP'
'SUP-INF.'    7   'K'  'MAX'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'AREFFIX'
'SUP-OUT.'    8   'K'  'MIN'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'EXTRAP'
'SUP-INF.'    8   'K'  'MAX'  'I'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0     'AREFFIX'
'CUT NAME    BLK  FACE   PLACE DIREC1 BEGIN   END  DIREC2 BEGIN   END  IN-ORDER'
'CUT_1'       3   'J'  'MAX'  'K'   'MIN'  'MAX'  'I'   'MIN'  'MAX'    0
'CUT_1'       1   'J'  'MAX'  'K'   'MIN'  'MAX'  'I'   'MAX'  'MIN'    0
'CUT_2'       4   'J'  'MIN'  'K'   'MIN'  'MAX'  'I'   'MIN'  'MAX'    0
'CUT_2'       3   'J'  'MIN'  'K'   'MIN'  'MAX'  'I'   'MAX'  'MIN'    0
'CUT_3'       4   'J'  'MAX'  'K'   'MIN'  'MAX'  'I'   'MIN'  'MAX'    0
'CUT_3'       2   'J'  'MAX'  'K'   'MIN'  'MAX'  'I'   'MAX'  'MIN'    0
'CUT_4'       5   'J'  'MAX'  'K'   'MIN'  'MAX'  'I'   'MIN'  'MAX'    0
'CUT_4'       1   'J'  'MIN'  'K'   'MIN'  'MAX'  'I'   'MIN'  'MAX'    0
'CUT_5'       6   'I'  'MIN'  'J'   'MIN'  'MAX'  'K'   'MIN'  'MAX'    0
'CUT_5'       3   'I'  'MAX'  'J'   'MIN'  'MAX'  'K'   'MIN'  'MAX'    0
'CUT_6'       6   'I'  'MAX'  'J'   'MIN'  'MAX'  'K'   'MIN'  'MAX'    0
'CUT_6'       5   'I'  'MIN'  'J'   'MIN'  'MAX'  'K'   'MIN'  'MAX'    0
'CUT_7'       6   'J'  'MAX'  'K'   'MIN'  'MAX'  'I'   'MIN'  'MAX'    0
'CUT_7'       1   'I'  'MIN'  'K'   'MIN'  'MAX'  'J'   'MAX'  'MIN'    0
'CUT_8'       7   'J'  'MIN'  'K'   'MIN'  'MAX'  'I'   'MIN'  'MAX'    0
'CUT_8'       5   'J'  'MIN'  'K'   'MIN'  'MAX'  'I'   'MAX'  'MIN'    0
'CUT_9'       7   'J'  'MAX'  'K'   'MIN'  'MAX'  'I'   'MIN'  'MAX'    0
'CUT_9'       2   'J'  'MIN'  'K'   'MIN'  'MAX'  'I'   'MIN'  'MAX'    0
'CUT_10'      8   'I'  'MIN'  'J'   'MIN'  'MAX'  'K'   'MIN'  'MAX'    0
'CUT_10'      7   'I'  'MAX'  'J'   'MIN'  'MAX'  'K'   'MIN'  'MAX'    0
'CUT_11'      8   'J'  'MIN'  'K'   'MIN'  'MAX'  'I'   'MIN'  'MAX'    0
'CUT_11'      6   'J'  'MIN'  'K'   'MIN'  'MAX'  'I'   'MAX'  'MIN'    0
'CUT_12'      8   'I'  'MAX'  'J'   'MIN'  'MAX'  'K'   'MIN'  'MAX'    0
'CUT_12'      4   'I'  'MIN'  'J'   'MIN'  'MAX'  'K'   'MIN'  'MAX'    0
'CUT_13'      8   'J'  'MAX'  'K'   'MIN'  'MAX'  'I'   'MIN'  'MAX'    0
'CUT_13'      2   'I'  'MAX'  'K'   'MIN'  'MAX'  'J'   'MIN'  'MAX'    0
```

Appendix 1.C, Driver Script for Fuel Injector Analysis

Driver script written in csh
Plans to convert to PERL

```
#!/bin/csh -f
# Sample simulator to Dakota system call script
# See User Manual for instructions
#
# bvbw 10/24/01
#
# modified for use with Vulcan
#

# $argv[1] is params.in.(fn_eval_num) FROM Dakota
# $argv[2] is results.out.(fn_eval_num) returned to Dakota

# ------------------------
# Set up working directory
# ------------------------

set num = `echo $argv[1] | cut -c 11-`
```

Get the unique number for this run

```
cp -r template_dir workdir.$num
mv $argv[1] workdir.$num
cd workdir.$num
```

Copy the template, or baseline directory, to the working directory

```
# --------------
# PRE-PROCESSING
# --------------
# Use the following line if SNL's APREPRO utility is used instead
# of transfer_perl.
# ../aprepro -c '*' -q --nowarning ros.template ros.in

../dprepro $argv[1] template.fra _az.fra
../dprepro $argv[1] cap_surf_inp.template cap_surf.inp

../dprepro $argv[1] vulcan.template vulcan.inp
```

Use DPREPRO utility to read PARAMS.IN.X file (generated by DAKOTA), parse data then search and replace template files with data

```
# --------
# ANALYSIS
# --------
# Grid Generation
# 1st generate the capsurface
../capSurface cap_surf.inp
```

Generate the cap surface for the fuel injector tube

```
# now generate the new grid
Ggrid _az.fra -D 2 > _az.out
mrgb blk.tmp -cr > mrgb.out
chfmt blk.tmp.tmp -f p3d -D > chfmt.out
```

Generate the grid – schedule (_az.sch) file copied from template directory

Merge the blocks in the grid – merge schedule can be in template directory

```
# RUN VULCAN
set qscript = qv_sgopt_pga$num
cp qv_fi $qscript

#set qnum = `qsub $qscript | cut -c -3`
qsub $qscript | cut -c -3 > QNUM
set qnum = `cat QNUM`
```

Submit VULCAN job to PBS que

```
# now wait until the job has finished by
# checking for the existance of the $qscript.o$qnum file

while (1)
sleep 10
  if (-f $qscript.o$qnum)  then  # comment
    break
  endif
end
```

Loop until output file exists, indicating the VULCAN has completed or erred out

```
# ---------------
# POST-PROCESSING
# ---------------
# for now use tail and hard code position in the file -
# plan to make more general later
tail -n 1 vulcan.ifam_his.tec_1 | cut -c 140- >! $argv[2]
```

Read the VULCAN FOM file and pipe the appropriate value into the RESULTS.OUT.X file, argument 2 into the script

```
# NOTE: moving $argv[2] at the end of the script avoids any
# problems with read race conditions.
mv $argv[2] ../.
# --------
# Clean up
# --------
cd ..
```

Move the output file up 1 directory
CD up 1 directory to start the next simulation

REFFERENCES

[1] White, J.A., and Morrison, J.H., "A Pseudo-Temporal Multi-Grid Relaxation Scheme for Solving the Parabolized Navier-Stokes Equations," AIAA Paper No. 99-3360, June 1999. http://vulcan-cfd.larc.nasa.gov/WebPage/aiaa99-3360.pdf

[2] Eldred, M.. S., et. al.., "DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification and Sensitivity Analysis. Version 3.1 Users Manual", April, 2003. http://endo.sandia.gov/DAKOTA

[3] Papadopoulos , P. M., Katz, M. J. and Bruno, G., "NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters", October 2001, Cluster 2001.

[4] Eiseman, P et. al., "User's Guide and Reference Manual for TIL, Ggrid and Other Utilities", August 3, 1999. http://www.gridpro.com